CSRN 2601

(Eds.)

- **Zhigeng Pan**
  Hangzhou Normal University, Hangzhou, China
- **Vaclav Skala**
  University of West Bohemia, Czech Republic

Computer Science Research Notes

24th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision
WSCG 2016
Plzen, Czech Republic
May 30 – June 3, 2016

Proceedings

WSCG 2016

Full Papers Proceedings

ISSN 2464–4617 (print) ISSN 2464–4625 (CD-ROM)

ISSN 2464–4617 (print) ISSN 2464–4625 (CD-ROM)

CSRN 2601

(Eds.)

- **Zhigeng Pan**
  Hangzhou Normal University, Hangzhou, China
- **Vaclav Skala**
  University of West Bohemia, Czech Republic

Computer Science Research Notes

24th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision
WSCG 2016
Plzen, Czech Republic
May 30 – June 3, 2016

Proceedings

WSCG 2016

Full Papers Proceedings

Vaclav Skala – Union Agency

ISSN 2464–4617 (print) ISSN 2464–4625 (CD-ROM)

Computer Science Research Notes
CSRN 2601

Editor-in-Chief: Vaclav Skala
c/o University of West Bohemia
Univerzitni 8
CZ 306 14 Plzen
Czech Republic
skala@kiv.zcu.cz *http://www.VaclavSkala.eu*

Managing Editor: Vaclav Skala

Publisher & Author Service Department & Distribution:
Vaclav Skala - UNION Agency
Na Mazinach 9
CZ 322 00 Plzen
Czech Republic
Reg.No. (ICO) 416 82 459

ISSN 2464-4617 (Print) **ISSN 2464-4625** (CD/DVD)
ISBN 2464-4625 (CD ROM**)** **ISBN 978-80-86943-57-2** (CD/-ROM)

WSCG 2016
International Program Committee

Benger,Werner (United States)
Bilbao,Javier,J. (Spain)
Bittner,Jiri (Czech Republic)
Buehler,Katja (Austria)
Chaudhuri,Debasis (India)
Chmielewski,Leszek (Poland)
Coquillart,Sabine (France)
Daniel,Marc (France)
de Geus,Klaus (Brazil)
Dingliana,John (Ireland)
Drechsler,Klaus (Germany)
Durikovic,Roman (Slovakia)
Falcidieno,Bianca (Italy)
Feito,Francisco (Spain)
Ferguson,Stuart (United Kingdom)
Flaquer,Juan (Spain)
Garcia Hernandez,Ruben Jesus (Germany)
Giannini,Franca (Italy)
Gudukbay,Ugur (Turkey)
Guthe,Michael (Germany)
Jeschke,Stefan (Austria)
Juan,M.-Carmen (Spain)
Kim,H. (Korea)
Max,Nelson (United States)
Molla,Ramon (Spain)
Montrucchio,Bartolomeo (Italy)
Muller,Heinrich (Germany)
Murtagh,Fionn (United Kingdom)
Pan,Rongjiang (China)
Pedrini,Helio (Brazil)
Platis,Nikos (Greece)
Renaud,christophe (France)
Richardson,John (United States)
Ritter,Marcel (Austria)
Rojas-Sola,Jose Ignacio (Spain)
Sanna,Andrea (Italy)
Segura,Rafael (Spain)
Skala,Vaclav (Czech Republic)
Slavik,Pavel (Czech Republic)
Sousa,A.Augusto (Portugal)
Szecsi,Laszlo (Hungary)
Teschner,Matthias (Germany)
Tokuta,Alade (United States)
Trapp,Matthias (Germany)
Viola,Ivan (Austria)
Wu,Shin-Ting (Brazil)
Wuensche,Burkhard,C. (New Zealand)
Wuethrich,Charles (Germany)
Zara,Jiri (Czech Republic)
Zemcik,Pavel (Czech Republic)

WSCG 2016
Board of Reviewers

Abad,Francisco (Spain)
Adzhiev,Valery (United Kingdom)
Agathos,Alexander (Romania)
Ahmad,Khurshid (Ireland)
Akleman,Ergun (United States)
Amditis,Angelos (Greece)
Ammi,Mehdi (France)
Ammi,Mehdi (France)
Ariu,Davide (Italy)
Assarsson,Ulf (Sweden)
Aveneau,Lilian (France)
Ayala,Dolors (Spain)
Backfrieder,Werner (Austria)
Barthe,Loic (France)
Battiato,Sebastiano (Italy)
Baum,David (Germany)
Benes,Bedrich (United States)
Benger,Werner (United States)
Benoit,Crespin (France)
Biasotti,Silvia (Italy)
Bilbao,Javier,J. (Spain)
Biri,Venceslas (France)
Birra,Fernando (Portugal)
Bittner,Jiri (Czech Republic)
Bosch,Carles (Spain)
Bouatouch,Kadi (France)
Boukaz,Saida (France)
Bourdin,Jean-Jacques (France)
Bourke,Paul (Australia)
Bouville,Christian (France)
Bruckner,Stefan (Austria)
Bruder,Gerd (Germany)
Brun,Anders (Sweden)
Bruni,Vittoria (Italy)
Brunnett,Guido (Germany)
Buehler,Katja (Austria)
Bulo,Samuel Rota (Italy)
Buriol,Tiago Martinuzzi (Brazil)
Cakmak,Hueseyin (Germany)
Camahort,Emilio (Spain)
Casciola,Giulio (Italy)
Chaine,Raphaelle (France)
Chaudhuri,Debasis (India)
Chen,Falai (China)
Chmielewski,Leszek (Poland)
Choi,Sunghee (Korea)
Chover,Miguel (Spain)
Chrysanthou,Yiorgos (Cyprus)
Chuang,Yung-Yu (Taiwan)
Cline,David (United States)
Coquillart,Sabine (France)
Corcoran,Andrew (Ireland)
Cosker,Darren (United Kingdom)
Daniel,Marc (France)
Daniels,Karen (United States)
de Amicis,raffaele (Italy)
de Geus,Klaus (Brazil)
de Oliveira Neto,Manuel Menezes (Brazil)
De Paolis,Lucio Tommaso (Italy)
Debelov,Victor (Russia)
Dingliana,John (Ireland)
Doellner,Juergen (Germany)
Dokken,Tor (Norway)
Drechsler,Klaus (Germany)
Durikovic,Roman (Slovakia)
Eisemann,Martin (Germany)
Erleben,Kenny (Denmark)
Falcidieno,Bianca (Italy)
Faudot,Dominique (France)
Feito,Francisco (Spain)
Ferguson,Stuart (United Kingdom)
Fiorentino,Michele (Italy)
Flaquer,Juan (Spain)
Fuenfzig,Christoph (Germany)
Gain,James (South Africa)
Galo,Mauricio (Brazil)
Garcia Hernandez,Ruben Jesus (Germany)
Garcia-Alonso,Alejandro (Spain)
Gavrilova,M. (Canada)
Gianelli,Carlota (Germany)
Giannini,Franca (Italy)

Gobron,Stephane (Switzerland)
Goebel,Martin (Germany)
Gonzalez,Pascual (Spain)
Grau,Sergi (Spain)
Gu,Xianfeng (United States)
GuĂŠrin,Eric (France)
Gudukbay,Ugur (Turkey)
Guthe,Michael (Germany)
Habel,Ralf (Switzerland)
Hall,Peter (United Kingdom)
Hansford,Dianne (United States)
Haro,Antonio (United States)
Hast,Anders (Sweden)
Hauser,Helwig (Norway)
Havemann,Sven (Austria)
Havran,Vlastimil (Czech Republic)
Hege,Hans-Christian (Germany)
Hernandez,Benjamin (United States)
Herout,Adam (Czech Republic)
Hicks,Yulia (United Kingdom)
Hildenbrand,Dietmar (Germany)
Hinkenjann,Andre (Germany)
Horain,Patrick (France)
Horain,Patrick (France)
House,Donald (United States)
Ihrke,Ivo (Germany)
Iwasaki,Kei (Japan)
Jeschke,Stefan (Austria)
Jiang,Jianmin (China)
Jones,Mark (United Kingdom)
Juan,M.-Carmen (Spain)
Juettler,Bert (Austria)
Kanai,Takashi (Japan)
Kim,H. (Korea)
Klosowski,James (United States)
Kohout,Josef (Czech Republic)
Kolcun,Alexej (Czech Republic)
Krueger,Jens (Germany)
Kumar,Subodh (India)
Kurillo,Gregorij (United States)
Kurt,Murat (Turkey)
Kyratzi,Sofia (Greece)
Lanquentin,Sandrine (France)
Larboulette,Caroline (France)
Lee,Jong Kwan Jake (United States)
Lengyel,Eric (United States)
Lien,Jyh-Ming (United States)
Lindow,Norbert (Germany)
Liu,SG (China)
Liu,Damon Shing-Min (Taiwan)
Lopes,Adriano (Portugal)
Loscos,Celine (France)
Lucas,Laurent (France)
Lutteroth,Christof (New Zealand)
Maciel,Anderson (Brazil)
Maddock,Steve (United Kingdom)
Magnor,Marcus (Germany)
Manak,Martin (Czech Republic)
Mandl,Thomas (Germany)
Manzke,Michael (Ireland)
Mas,Albert (Spain)
Masia,Belen (Spain)
Masood,Syed Zain (United States)
Matey,Luis (Spain)
Matkovic,Kresimir (Austria)
Max,Nelson (United States)
McDonnell,Rachel (Ireland)
McKisic,Kyle (United States)
Meng,Weiliang (China)
Mestre,Daniel,R. (France)
Metodiev,Nikolay Metodiev (United States)
Meyer,Alexandre (France)
Mokhtari,Marielle (Canada)
Molina Masso,Jose Pascual (Spain)
Molla,Ramon (Spain)
Montrucchio,Bartolomeo (Italy)
Morigi,Serena (Italy)
Muller,Heinrich (Germany)
Murtagh,Fionn (United Kingdom)
Myszkowski,Karol (Germany)
Niemann,Henrich (Germany)
Nishita,Tomoyuki (Japan)
Okabe,Makoto (Japan)
Oliveira, Jr.,Pedro Paulo (Brazil)
Oyarzun Laura,Cristina (Germany)
Pala,Pietro (Italy)
Pan,Rongjiang (China)
Papaioannou,Georgios (Greece)
Paquette,Eric (Canada)
Pasko,Alexander (United Kingdom)
Pasko,Galina (United Kingdom)

Pastor,Luis (Spain)
Patane,Giuseppe (Italy)
Patow,Gustavo (Spain)
Pedrini,Helio (Brazil)
Pereira,Joao Madeiras (Portugal)
Perret,Jerome (France)
Peters,Jorg (United States)
Pettre,Julien (France)
Peytavie,Adrien (France)
Pina,Jose Luis (Spain)
Platis,Nikos (Greece)
Plemenos,Dimitri (France)
Post,Frits,H. (Netherlands)
Poulin,Pierre (Canada)
Praktikakis,Ioannis (Greece)
Puig,Anna (Spain)
Puppo,Enrico (Italy)
Rafferty,Karen (United Kingdom)
Reisner-Kollmann,Irene (Austria)
Renaud,christophe (France)
Renaud,christophe (France)
Reyes-Lecuona,Arcadio (Spain)
Richardson,John (United States)
Ritschel,Tobias (Germany)
Ritter,Marcel (Austria)
Rojas-Sola,Jose Ignacio (Spain)
Rokita,Przemyslaw (Poland)
Runde,Christoph (Germany)
Ruther,Heinz (South Africa)
Sacco,Marco (Italy)
Sadlo,Filip (Germany)
Sakas,Georgios (Germany)
Salvetti,Ovidio (Italy)
Sanna,Andrea (Italy)
Santos,Luis Paulo (Portugal)
Sapidis,Nickolas,S. (Greece)
Savchenko,Vladimir (Japan)
Schultz,Thomas (Germany)
Schumann,Heidrun (Germany)
Segura,Rafael (Spain)
Seipel,Stefan (Sweden)
Sellent,Anita (Switzerland)
Shesh,Amit (United States)
Sik-Lanyi,Cecilia (Hungary)
Slavik,Pavel (Czech Republic)
Sochor,Jiri (Czech Republic)
Solis,Ana Luisa (Mexico)
Sommer,Bj?rn (Germany)
Sourin,Alexei (Singapore)
Sousa,A.Augusto (Portugal)
Sramek,Milos (Austria)
Sreng,Jean (France)
Staadt,Oliver (Germany)
Stricker,Didier (Germany)
Stroud,Ian (Switzerland)
Subsol,Gerard (France)
Suescun,Angel ()
Sunar,Mohd-Shahrizal (Malaysia)
Sundstedt,Veronica (Sweden)
Svoboda,Tomas (Czech Republic)
Szecsi,Laszlo (Hungary)
Tang,Min (China)
Tang,Qian (China)
Taubin,Gabriel (United States)
Tavares,Joao Manuel R.S. (Portugal)
Teschner,Matthias (Germany)
Theussl,Thomas (Saudi Arabia)
Tian,Feng (United Kingdom)
Tobler,Robert (Austria)
Todt,Eduardo (Brazil)
Tokuta,Alade (United States)
Torrens,Francisco (Spain)
Trapp,Matthias (Germany)
Triantafyllidis,Georgios (Greece)
Tytkowski,Krzysztof (Poland)
Umlauf,Georg (Germany)
Vanderhaeghe,David (France)
Vasa,Libor (Czech Republic)
Vazquez,Pere-Pau (Spain)
Vazquez,Pere Pau (United States)
Viola,Ivan (Austria)
Vitulano,Domenico (Italy)
Vosinakis,Spyros (Greece)
Walczak,Krzysztof (Poland)
WAN,Liang (China)
Wang,Charlie,C.L. (Hong Kong SAR)
Weber,Andreas (Germany)
Wenger,Raphael (United States)
Westermann,Ruediger (Germany)
Wu,Enhua (China)
Wu,Shin-Ting (Brazil)
Wuensche,Burkhard,C. (New Zealand)

Wuethrich,Charles (Germany)
Xin,Shi-Qing (Singapore)
Yoshizawa,Shin (Japan)
YU,Qizhi (United Kingdom)
Yue,Yonghao (Japan)
Zachmann,Gabriel (Germany)
Zalik,Borut (Slovenia)
Zara,Jiri (Czech Republic)
Zemcik,Pavel (Czech Republic)
Zhang,Xiaopeng (China)
Zhang,Xinyu (United States)
Zhu,Ying (United States)
Zillich,Michael (Austria)
Zitova,Barbara (Czech Republic)
Zwettler,Gerald (Austria)

WSCG 2016

Full Papers Proceedings

Contents

ISSN 2464-4617 (print) WSCG 2016 - 24th Conference on Computer Graphics, Visualization and Computer Vision 2016
ISSN 2464-4625 (CD-ROM)

Using Trajectories derived by Dense Optical Flows as a Spatial Component in Background Subtraction

Martin Radolko
University of Rostock
and Fraunhofer IGD
Joachim-Jungius 11
Rostock 18059
martin.radolko@igd-r.fraunhofer.de

Fahimeh Farhadifard
University of Rostock
and Fraunhofer IGD
Joachim-Jungius 11
Rostock 18059
fahimeh.farhadifard@igd-r.fraunhofer.de

ABSTRACT

Foreground-Background Segregation has been intensively researched in the last decades as it is an important first step in many Computer Vision tasks. Nonetheless, there are still many open questions in this area and in this paper we focus on a special surveillance scenario where a static camera monitors a predefined region. This restrain makes some aspects easier and good results could be achieved with Background Subtraction methods. However, these only work pixelwise and lack the spatial component completely. We suggest an approach to add the crucial spatial information to the segmentations with Dense Optical Flows. For this, a number of successive images are taken from the video to compute the Trajectories of the pixels through these frames. This enables us to fuse the information from the several images and use this for segmentation. The algorithm was evaluated on a video from a surveillance camera and showed promising results.

Keywords

Foreground-Background Segregation, Background Subtraction, Dense Optical Flows, Video Segmentation

1 INTRODUCTION

Detecting objects of interest in an image or video is an arduous task which has distressed the computer vision community for decades. The abundance of different circumstances and aims (definitions of objects of interest) makes it impossible to create one general solution to this problem. Hence, different special areas have developed over time like the recognition of a few previously specified objects in an image or video [Duraisamy and Jane, 2014], general object detection for images [de Carvalho et al., 2010] or detection of moving objects in a video [El Harrouss et al., 2015].

In this paper we consider the last case and want to detect all moving objects in a video scene. Therefore, we assume a video from a static camera, e.g. a surveillance camera mounted in a shopping street. This makes it easier to detect the movement of objects because the background (buildings, streets etc.) are completely stable in their position over time. There are some attempts to extend these techniques to non-static cameras by estimating the three dimensional trajectories of the background[Sheikh et al., 2009]. However, the creation of a meaningful background model is error-prone because small imperfections in the trajectories will accumulate over time and thwart any precise modeling. That's why, the trajectories themselves were only used as cues and everything with a different trajectory than the background was labeled as foreground.

If the video is taken with a static camera, the background can be modeled over time with statistical methods. Moving objects, which are the interesting objects to most users, can then be detected via Background Subtraction. This is a very popular approach since it gives State of the Art results for one of the main use cases of segmentation algorithms, static surveillance cameras. Also, it is very effective and can be done in real time[Tabkhi et al., 2015]. However, there are two issues with this methods which have to be addressed.

The first one is the background modeling itself, which is affected by the presence of the ubiquitous noise in images. This noise can be reduced by applying a Gaussian Filter on the image but slight variations are always present (otherwise too much information would be lost due to the strong Gaussian smoothing) and have to be compensated with an adequate threshold, ideally one which adapts to the specific scene and camera [Soeleman et al., 2012]. Furthermore, the background can change over time, e.g. a car that parks or

light which is turned on or off. To model these occurrences an accurate adjustment of the learning rate of the model is required and sometimes an event detection, which changes the learning rate or resets the model for special cases, can be beneficial [Radolko et al., 2015, Toyama et al., 1999].

The second issue to be addressed is the pixelwise thresholding and labeling in the Background Subtraction phase which incorporates no spatial component. Images are usually quite smooth with only few distinctive edges. Without a spatial model, this smoothness cannot be represented by the method and no coherent objects will be segmented but many small fragments. There are a vast number of models which have been applied on Background Subtraction results to add spatial information. One example is Graph Cut [Tang and Miao, 2008], which finds the best cut in a probability map created by the Background Subtraction. Another method is the Markov Random Field which models direct neighbourhood relations between single pixels and therefore enforces a spatial coherency [Wan and Wang, 2010].

In this paper, we suggest a different approach of addressing this lack of spatial information of the Background Subtraction method. We use Dense Optical Flows to acquire spatial-temporal information about each pixel, which allows us to track pixels through several frames of the video. Therefore, the data from several others frames of the video can be used to enhance the segmentation, which results in a less noise-sensitive algorithm. Also, the Dense Optical Flow gives us a second cue for the foreground detection. High flows correspond to moving objects in the video, which should be detected by the Background Subtraction. Thereby, the optical flow can verify and improve the Background Subtraction result in two different ways.

2 APPROACH

The aim of our approach is to detect all objects of interest in a video. The camera is assumed to be static to simplify the task and consequently the areas of interest are defined as all objects in motion. If immobile objects are of interest it would be meaningless to make a video with a static camera, a single picture would be enough. First we will shortly introduce the Background Subtraction method and the algorithm to estimate the Dense Optical Flow of the video. In the next step the data from the Dense Optical Flow is used to estimate the position of an object in the previous and future frames of the video. This is used to unify succesive segmentations, which increases the reliability of the object detection.

2.1 Background Subtraction

To get a first estimation about the moving objects in the scene, we use the Gaussian Switch Model Background Subtraction described in [Radolko and Gutzeit, 2015]. There are exactly two pixelwise Background models trained over time and both use a running Gaussian estimation. However, only one is completely updated with every new frame and the other only partially for those pixels which are classified as background.

The model which gets only partially updated usually gives a better background estimation because foreground objects are omitted in the updating process and do not corrupt the model. However, since the foreground-background classification is based on the same model it works sometimes like a self-fulfilling prophecy and cannot adapt to some special cases. Therefore, the second model, which is completely updated with every frame, is necessary. A comparison between both models allows us to detect these special cases and then switch to the appropriate model. For more details see [Radolko and Gutzeit, 2015].

To further process these information simple foreground-background labels are not sufficient. Therefore, we calculate the probability of being in the foreground for each pixel. For every color channel of the image the Euclidean distance between the values of the current frame and the model is taken and weighted with the variance of the Gaussian model for that channel and pixel. To unify these to a single probability measure, the three values are scaled, bounded (to avoid probabilities larger than one) and added together. This gives us a continuous probability value, which carries much more information than a simple binary label.

2.2 Dense Optical Flow

The dense optical flow is computed with the algorithm described in [Farnebäck, 2003]. The first step in creating the Optical Flow between two images consists of modeling both images as Polynomial Expansions. By solving a system of linear Equations, a displacement vector between these two models can be computed for each pixel location. However, single pixels are too vulnerable to noise, so that no smooth or accurate solution is attainable in this way. This issue can be resolved by using small areas instead of single pixels. We always use patches with the size of 15×15 pixels and radially decreasing weights for the computation of the displacement vectors in this paper.

A priori knowledge can be incorporated into the computation by setting the starting values accordingly. In this case, the algorithm only has to compute the displacement vector to the a priori knowledge, which is usually much smaller than the raw displacement between the two images. This is very beneficial because the calculation of large displacement vectors is prone to errors. This approach can be used in two ways to improve the Optical Flows. Since we use a video and want to compute the Dense Optical Flows for all frames, the

Figure 1: In the top row the intensities of the optical flow are depicted for an image. On the left side for the right-left movement and on the right for the up-down movement. In the bottom row are the orignal frame from the video and the segmentation result.

flow from the last frame can be used as a good starting approximation for the current frame. In the second strategy a hierarchical approach is used and the Optical Flow is calculated first on a low resolution version of the image. The result of this calculation is then used as a start value for versions with higher resolution.

Thereby, the Optical Flows can be computed faster and in a more stable way over the course of a video. An example of the results of the optical flow can be seen in Figure 1. It can clearly be seen that optical flow better represents the smoothness in natural images as a background subtraction method and is therefore in itself an important second cue for the segmentation. In the next step these extracted information are used to enhance the segmentations derived from the Background Subtraction.

2.3 Using Optical Flows as a Spatial Component

Background Subtraction algorithms usually give good segmentation results but suffer from the lack of spatial information incorporated. Therefore, false detections of single pixels or small areas are common and prevent the precise recognition of smooth and coherent foreground objects. We try to mitigate this behavior by calculating the pathway of each pixel over several frames (past and future) and smooth the foreground probability with a Gaussian filter over this pathway.

The pathway of each pixel can be easily derived from the Dense Optical Flows. Here they are denoted as

$$DOF^x_{n-1,n}(x,y) \quad \text{and} \quad DOF^y_{n-1,n}(x,y). \tag{1}$$

The Optical Flow is here computed between the $n-1$-th and n-th frame of the video for the pixel at location (x,y) in frame $n-1$. DOF^x denotes the horizontal flow and accordingly DOF^y the vertical flow. Thereby, the pixel $p=(x,y)$ in frame $n-1$ has moved to location

$$\tilde{x} = x - DOF^x_{n-1,n}(x,y) \tag{2}$$
$$\tilde{y} = y - DOF^y_{n-1,n}(x,y). \tag{3}$$

in frame n according to the Optical Flow. The new location for p in the frames $n-2$ or $n+1$ can be calculated in a similar way. With this method pathways for single pixels can be computed over several frames. However, since the Optical Flows are not perfectly accurate and errors accumulate radically over longer pathways, reliable information can only be derived for a small number of past or future frames. In our approach, the pathways for 7 frames ($n-3$ to $n+3$) are computed and used to enhance the foreground probabilities of the n-th frame. This is done by using a Gaussian filter along the pathway centered at the n-th frame with standard deviation of 0.75. The low standard deviation ensures that the importance of the values along the pathway decreases rapidly in both directions, this corresponds to the rapidly decreasing certainty of the correctness of the pathway.

Furthermore, the Optical Flows are an indicator of movement in the scene themselves and can be used as a foreground cue. For this purpose, the location of the pixel p in the frames $n-1$ and n are taken and the Euclidean distance $d_p^{n-1,n}$ is computed. The same is done for the locations in the frames n and $n+1$. The foreground probability w^p of pixel p derived from the

Background Subtraction is taken and enhanced in the following way

$$\tilde{w}^p = \frac{2}{3}w^p + \frac{1}{6}\min(d^p_{n-1,n}, 1) + \frac{1}{6}\min(d^p_{n,n+1}, 1). \quad (4)$$

The weights of the different components have determined experimentally, only condition is that they add up to one to ensure that the alue stays in the range of $[0,1]$.

The third step is a slight spatial smoothing of each probability map separately in which the differences of the value of the central pixel with that of all other pixels in a 3×3 neighborhood are summed up, weighted and the result is added to the probability value of the central pixel. These three steps; the smoothing over the trajectories, the adding of the Optical Flow prior and the spatial smoothing; are applied successively on a batch of segmentations derived from the Background Subtraction. For the tests in this paper a batch size of 100 frames was used.

We iterated several times over the whole batch and applied all three methods each time. This elaborate process is useful because the changes in the segmentation in the first step influence the smoothing over the trajectories in the next steps. The quality and smoothness of the probability maps increases gradually over the iterations and with this also the trajectories provide better information in each step and thereby improve the segmentations further. As the trajectories cannot be fully computed for the first and last three frames of the batch, the segmentations are only computed for the frames 4 to 97 of the batch. Some trajectories cannot be computed because for each trajectory we have to reach three frames into the future and past, but for the first and last three frames of the batch this is obviously not possible.

This process, of course, needs a termination criteria which stops the iteration. Instead of a fixed number of iterations, we measured the changes in the probability fields from one iteration to the next, and if these changes where smaller than a specified threshold the loop would break. To reduce the computational cost we constrained this measurement to the probability map for one frame of the batch. A summary of the whole segmentation process can be seen in Algorithm 1.

Algorithm 1 Our Method

```
** initial segmentations **
for i=1:100 do
    frames[i] ← getimage(video)
    UpdateBackgroundModel(frames[i])
    segs[i] ← BackgroundSubtraction(frames[i])
**get  Dense Optical Flows**
for i=1:99 do
    DOF[i] = getDOF(frames[i], frames[i+1])
** get Trajectories **
Paths ← getTrajectories(DOF)
** use DOF and Paths to improve results **
while TerminationCriteria do
    for i=4:97 do
        segs[i] ← spatialsmoothing(segs[i])
        segs[i] ← AddDOFPrior(segs[i], DOF[i])
    for i=4:97 do
        segs[i] ← PathSmoothing(segs, Paths[i])
```

3 RESULTS

To test our method we used the Town Center Video [Benfold and Reid, 2011] which is made with a surveillance camera in a shopping street. The provided ground truth data is made for the comparison of tracking algorithms and not suitable for our purpose. Hence, we created several groundtruth picture ourselves manually to evaluate the algorithm. The ground truth data consists of a trimap which classifies each pixel as foreground (white), background (black) or uncertain (gray). Two examples of this can be seen in Figure 2, where also the results of our algorithm are depicted.

As comparison the algorithm from [Zivkovic, 2004] and [Zivkovic and Heijden, 2006] were used on the same data set. To have a neutral implementation of these algorithms we used that provided by OpenCV. The results of these two and our algorithm on the test images from Figure 2 can be seen in Table 1. The F1-Score and Matthews correlation coefficient (MCC) are taken as a measure for accuracy. In both measures our approach achieved a better result than the other algorithms and could significantly improve the results achieved by just using the GSM Background Subtraction.

All in all our approach shows good first results and is a promising novel method to add spatial information to pixel based segmentation methods. The addition of the optical greatly increases the robustness to camera noise as can be seen in the examples given and also helps with updating the background (when the background or light conditions changed) because there is a second cue to confirm or deny the existing model. The computation of the optical flows is done on the GPU and therefore very fast, nonetheless is the fusion of the data from the different methods too costly for a real time approach, it took us about one second per image for the computation.

4 CONCLUSIONS

We presented a new method to enhance the results of pixel based segmentation methods like Background Subtraction with spatial information, so that the results better reflect the smoothness of natural images. For this purpose we computed pixel trajectories over several frames of the video with Dense Optical Flows and

used this to fuse the data of several successive frames. This decreased the vulnerability of our segmentations to noise as the flaws in a single measurement (frame) could be corrected by the data obtained from the other frames. At the same time, this approach increased the smoothness of the segmentations because the foreground probabilities are smoothed over several frames and the smoothness of the Dense Optical Flows is also gradually transferred to the segmentation.

The usage of the trajectories derived from the Optical Flows to adjust the foreground probabilities for the pixels and the spatial smoothing based the differences of the probabilities in a 3×3 neighborhood further improved the results. In the future, we would like to develop a Dense Optical Flow algorithm which is especially designed for this approach and the circumstances. It should incorporate the knowledge about the static camera, so that large parts of the flow can be assumed to be zero. Also, to compute better trajectories we want to use several frames at once and compute the best trajectory over these instead of only computing the flow between two frames at a time. We hope to increase the robustness of the trajectories in this way and hence enlarge the benefit of our algorithm for the segmentations.

5 REFERENCES

[Benfold and Reid, 2011] Benfold, B. and Reid, I. (2011). Stable multi-target tracking in real-time surveillance video. In *CVPR*, pages 3457–3464.

[de Carvalho et al., 2010] de Carvalho, M., da Costa, A., Ferreira, A., and Marcondes Cesar Junior, R. (2010). Image segmentation using component tree and normalized cut. In *Graphics, Patterns and Images (SIBGRAPI), 2010 23rd SIBGRAPI Conference on*, pages 317–322.

[Duraisamy and Jane, 2014] Duraisamy, M. and Jane, F. (2014). cellular neural network based medical image segmentation using artificial bee colony algorithm. In *Green Computing Communication and Electrical Engineering (ICGCCEE), 2014 International Conference on*, pages 1–6.

[El Harrouss et al., 2015] El Harrouss, O., Moujahid, D., and Tairi, H. (2015). Motion detection based on the combining of the background subtraction and spatial color information. In *Intelligent Systems and Computer Vision (ISCV), 2015*, pages 1–4.

[Farnebäck, 2003] Farnebäck, G. (2003). Two-frame motion estimation based on polynomial expansion. In Bigun, J. and Gustavsson, T., editors, *Image Analysis*, volume 2749 of *Lecture Notes in Computer Science*, pages 363–370. Springer Berlin Heidelberg.

[Radolko et al., 2015] Radolko, M., Farhadifard, F., Gutzeit, E., and von Lukas, U. F. (2015). Real time video segmentation optimization with a modified normalized cut. In *Image and Signal Processing and Analysis (ISPA), 2015 9th International Symposium on*, pages 31–36.

[Radolko and Gutzeit, 2015] Radolko, M. and Gutzeit, E. (2015). Video segmentation via a gaussian switch background-model and higher order markov random fields. In *Proceedings of the 10th International Conference on Computer Vision Theory and Applications Volume 1*, pages 537–544.

[Sheikh et al., 2009] Sheikh, Y., Javed, O., and Kanade, T. (2009). Background subtraction for freely moving cameras. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 1219–1225.

[Soeleman et al., 2012] Soeleman, M., Hariadi, M., and Purnomo, M. (2012). Adaptive threshold for background subtraction in moving object detection using fuzzy c-means clustering. In *TENCON 2012 - 2012 IEEE Region 10 Conference*, pages 1–5.

[Tabkhi et al., 2015] Tabkhi, H., Sabbagh, M., and Schirner, G. (2015). An efficient architecture solution for low-power real-time background subtraction. In *Application-specific Systems, Architectures and Processors (ASAP), 2015 IEEE 26th International Conference on*, pages 218–225.

[Tang and Miao, 2008] Tang, Z. and Miao, Z. (2008). Fast background subtraction using improved gmm and graph cut. In *Image and Signal Processing, 2008. CISP '08. Congress on*, volume 4, pages 181–185.

[Toyama et al., 1999] Toyama, K., Krumm, J., Brumitt, B., and Meyers, B. (1999). Wallflower: Principles and practice of background maintenance. In *Seventh International Conference on Computer Vision*, pages 255–261. IEEE Computer Society Press.

[Wan and Wang, 2010] Wan, Q. and Wang, Y. (2010). Detecting moving objects by ant colony system in a map-mrf framework. In *E-Product E-Service and E-Entertainment (ICEEE), 2010 International Conference on*, pages 1–4.

[Zivkovic, 2004] Zivkovic, Z. (2004). Improved adaptive gaussian mixture model for background subtraction. In *Proceedings of the Pattern Recognition, 17th International Conference on (ICPR'04) Volume 2 - Volume 02*, ICPR '04, pages 28–31, Washington, DC, USA. IEEE Computer Society.

[Zivkovic and Heijden, 2006] Zivkovic, Z. and Heijden, F. (2006). Efficient adaptive density estimation

This research has been supported by the German Federal State of Mecklenburg-Western Pomerania and the European Social Fund under grants ESF/IV-BM-B35-0006/12 and V630-S-179- 2013/238.

left picture of Figure 1	[Zivkovic, 2004]	[Zivkovic and Heijden, 2006]	only GSM	our Approach
MCC	0.8324	0.8591	0.8543	0.8789
F1-Score	0.9899	0.9890	0.9894	0.9916

right picture of Figure 1	[Zivkovic, 2004]	[Zivkovic and Heijden, 2006]	only GSM	our Approach
MCC	0.8292	0.8598	0.7908	0.8937
F1-Score	0.9879	0.9899	0.9816	0.9930

Table 1: Evaluation of the segmentations shown in Figure 1 with two algorithms from Zivkovic for comparison.

per image pixel for the task of background subtraction. *Pattern Recogn. Lett.*, 27(7):773–780.

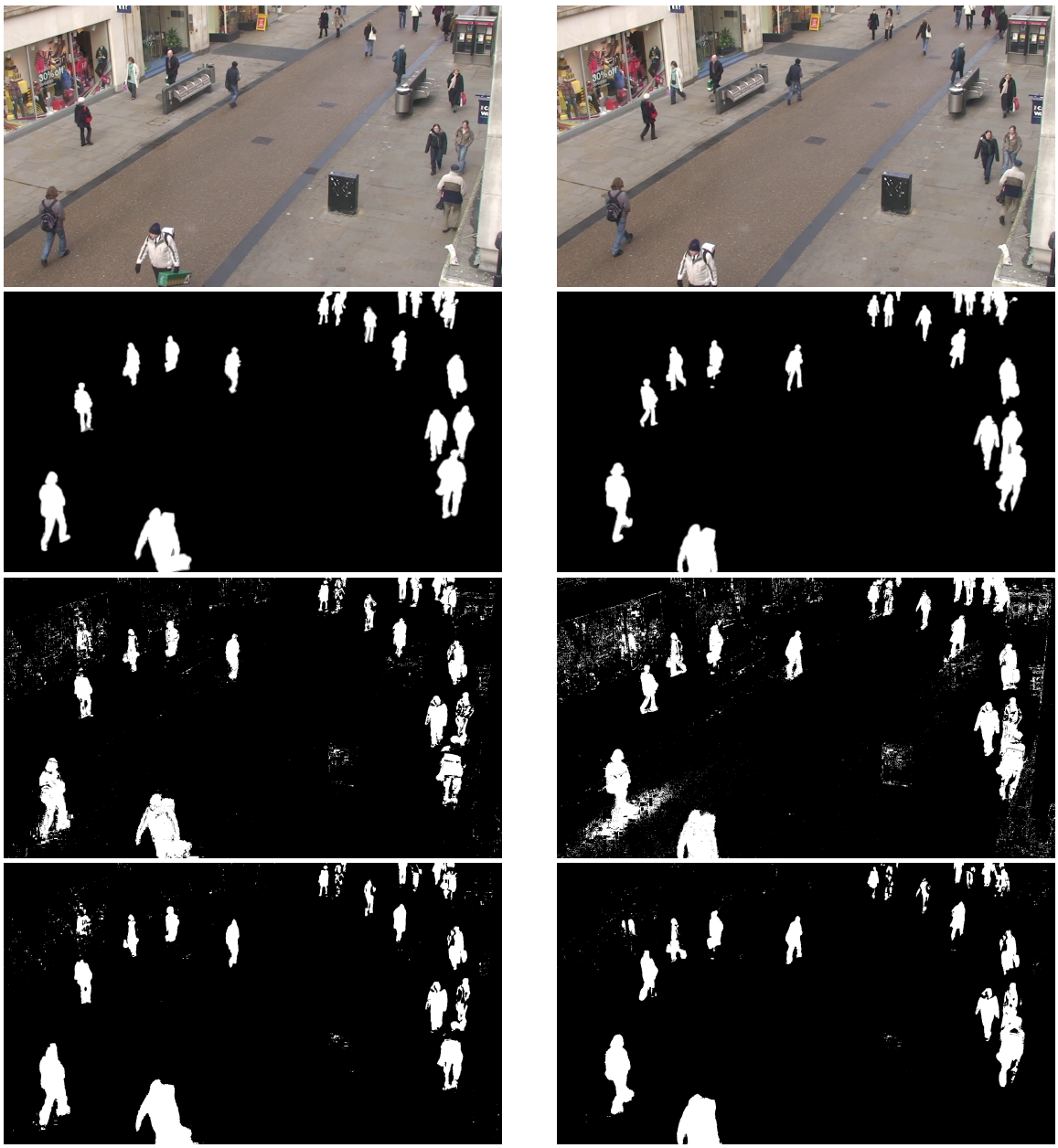


Figure 2: Shown are (top to bottom): original frame, ground truth, GSM Background Subtraction Segmentation, our result

ISSN 2464-4617 (print) WSCG 2016 - 24th Conference on Computer Graphics, Visualization and Computer Vision 2016
ISSN 2464-4625 (CD-ROM)

Identifying Linear Vector Fields on 2D Manifolds

Sebastian Volke
Leipzig University, Germany
volke@informatik.uni-leipzig.de

Stefan Koch
Leipzig University, Germany

Mario Hlawitschka
HTWK Leipzig, Germany

ABSTRACT

Local linearity of vector fields is a property that is well researched and understood. Linear approximation can be used to simplify algorithms or for data reduction. Whereas the concept is easy to implement in 2D and 3D, it loses meaning on manifolds as linearity has either to be defined based on an embedding in a higher-dimensional Cartesian space or on a map. We present an adaptive atlas-based vector field decomposition to solve the problem on manifolds and present its application on synthetic and climate data.

Keywords

Physical Sciences and Engineering, Environmental Sciences, Vector Field Projection, Vector Field Approximation

1 INTRODUCTION

Linear flow behavior in vector fields has been studied, because it allows an easy representation of the flow using the Jacobian matrix, and a characterization in few, well-understood basic linear flow patterns is possible [HSD04]. Investigations of linear vector fields exist for $\mathbb{R}^2$ and $\mathbb{R}^3$, and have proven to be a valuable tool for understanding flow. However, many real-world vector fields—e.g., in the geosciences (convection in the earth mantle), in the oceanic sciences (ocean currents), or in the atmospheric sciences (wind fields)—are tangential vector fields on curved surfaces. These datasets impose difficulties on the analysis, because tangent vectors can not directly be compared and the curvature of the surface has an influence on the vector field direction. Thus, linear flow behavior on manifolds is not directly describable and also not algorithmically accessible.

Usually, manifolds are only investigated at a small scale. One example is the use of cells that subdivide the manifold. A linear interpolation within the cell leads by definition to a linear field. If linearity is defined in the planar projection of each cell, the continuity across cell boundaries is not preserved across cell edges. This has implications, e.g., on the vector field's topology [AH11]. When extending the linearity across larger areas to approximate the field in what has been called a *Affine Linear Neighborhoods* [KWKH13] to describe the field around singularities, special care must be taken to avoid conflicts with the curvature of the surface.

In this paper, we propose a characterization of linear flow behavior on arbitrary two-dimensional manifolds and a method that allows the computation of linear neighborhoods on such manifolds. Based on this, we construct an atlas of the surface together with a corresponding, good linearly approximable, vector field representation. We demonstrate the application of the approach by evaluating synthetic fields on the sphere as well as on a real-world dataset of ocean currents.

2 RELATED WORK

The extraction and utilization of local linear vector field approximations has been discussed by several previous works. Schneider et al. [SRWS10] used the linear flow behavior in the vicinity of critical points to improve and accelerate stream surface integration. Wiebel et al. [WKS12] introduced glyphs to investigate the flow behavior at critical points and to study the interaction between them. Later, Koch et al. [KWKH13] introduced the *Affine Linear Neighborhood* (*ALN*). For a particular seed point, an ALN represents all connected points that can be approximated by an affine linear function while staying below a user-defined error threshold. They utilized their ALN definition to present a vector field approximation, as well as a compression, that is based on segmentation [KKW+15].So far, these methods only work on two- or three-dimensional fields but not on curved manifolds.

Other kinds of local approximation are the vector field moments introduced by Bujack et al. [BHSH14]. But there is no extension to arbitrary manifolds, yet. A different approach that lives entirely in the tangential space is the vector field definition using radial basis functions (e.g., Fuselier and Wright [FW09]). Here, the focus is on a reconstruction of the field from sample

points that are scattered on the surface. While this allows to locally characterize the field using kernels, no discussion of linear flow behavior has been done.

3 DEFINITION OF LINEARITY

Our goal is an algorithmically usable definition of linearity on manifolds. The main problem is that vectors at different points on the manifold live in different tangent spaces and are not directly comparable. Thus, it is unclear how linearity can be characterized. In the following, we formally define 2D manifolds and vector fields on them. Then, we give two possible definitions of linearity and discuss their applicability. By doing this, we want to make the notion of *"something that looks linear on the manifold"* accessible to computations.

3.1 2D Manifolds and Vector Fields

A *differentiable, two-dimensional manifold* M is a topological space that is locally homeomorphic to the two-dimensional Euclidean space. More specifically, it is defined by an *atlas*, i.e., a collection of *charts* $(U_i \subset M, \varphi_i : U_i \to B \subset \mathbb{R}^2)$, such that the U_i are an open cover of M (i.e. $\bigcup_i U_i = M$) and all φ_i are homeomorphisms. Furthermore, the chart transitions $\varphi_i \circ \varphi_j^{-1}$ need to be diffeomorphisms.

At each point $p \in M$, we can define a *tangent vector* as an equivalence class of differentiable curves $\gamma : (-\varepsilon, \varepsilon) \in M$ with $\gamma(0) = p$. Thereby, two curves γ_1 and γ_2 are equivalent, if and only if $(\varphi \circ \gamma_1)'(0) = (\varphi \circ \gamma_2)'(0)$ for any chart φ. These vectors form a two-dimensional vector space, the *tangent space* T_pM in p. The representation of a tangential vector v in chart φ is $(\varphi \circ v)'(0) \in \mathbb{R}^2$.

Note, that the tangent spaces T_pM and T_qM are distinct if $p \neq q$. The union of all tangent spaces forms the *tangent bundle* TM of M.

Now, we can define *vector fields* as functions that map a tangent vector to every point of M. A vector field v is a function $v : M \to TM$, such that $p \in M \mapsto v(p) \in T_pM$.

3.2 Polynomial Vector Fields

Many common two-dimensional manifolds are embeddable into $\mathbb{R}^3$, e.g., the sphere or the surface of mechanical components. Therefore, it is a natural idea to perform all calculations in the space of the embedding and to make use of polynomial vector fields in 3D (cf. [LP06]). A polynomial vector field on the manifold is then a 3D vector field restricted to the manifold that is everywhere tangential to the manifold.

If we can express a 3D vector field in the form of $v(x) = A \cdot x + b$, where $x, b \in \mathbb{R}^3$ and $A \in \mathbb{R}^{3\times 3}$ is the Jacobian matrix, we call it a linear vector field in 3D. Furthermore, we could consider the restriction to the manifold to be a linear vector field. For example, if the manifold is the unit sphere, the field given by $(-y, x, 0)$ would be such a linear polynomial field. However, the surface of some manifolds can not be described by a linear equation. In this case, a field that is linear on a geodesic on the surface may not be a linear polynomial vector field in 3D.

In general we would like to consider a field to be linear, if it has a constant Jacobian on the surface. But, there are two disadvantages with this approach: (1) It is unclear, how a Jacobian on a general manifold is defined. (2) If we simply use Jacobian matrices of the three-dimensional field, it is unclear how to interpret them and how to compare them for different positions of a manifold.

In any case, the Jacobian matrices would need to be transformed into a coordinate reference system that is tangential to the surface. Still, we have one remaining degree of freedom: the rotation around the normal vector of the tangential surface. A meaningful comparison of Jacobian matrices is only possible when consistent alignment is achieved. Therefore, this solution using polynomial vector fields is unsatisfactory for more complex manifolds than planes. Already on the sphere, a globally consistent alignment might be impossible according to the *Hairy ball theorem* [EG79].

3.3 Projected Vector Fields

Vectors on the manifold can not directly be compared and thus, linear vector fields can not be easily characterized within TM. However, linearity is well defined for vector fields on $\mathbb{R}^n$. We present an approach to describe linearity on 2D manifolds by using a projection of the surface into $\mathbb{R}^2$ and carrying over the vector field into this projection. For this, we assume that an atlas of the manifold is available.

Given a chart $\varphi : M \supset U \to B \subset \mathbb{R}^2, (u_1, u_2) \mapsto (x_1, x_2)$, the tangent space at every point $p \in M$ is characterized by the basis $\left\{ \frac{\partial}{\partial u_1}\Big|_p , \frac{\partial}{\partial u_2}\Big|_p \right\}$. Consequently, the vector field $v : M \to TM$ can be expressed as $v(p) = \sum_{i=1}^{2} a_i(p) \cdot \frac{\partial}{\partial u_i}\Big|_p$. Here, a_i are functions of the form $a_i : U \to \mathbb{R}$ and are called the *coefficients* of the vectors in the chart φ [Küh10]. So, for every chart we have a specific representation of the vector field in the form of coefficients. We can analyze these coefficients algorithmically if we find a way to make them comparable across different tangential spaces.

This can be done by moving all vectors into one tangential space by means of a *parallel transport*, i.e. [ZMT06]. In the general case, a computation of the parallel transport is difficult and expensive, as it requires geodesic curves to be integrated.

In this work, we chose a different approach and unify tangential spaces *by projecting them into a common*

space. Specifically, we project the basis vectors of each tangential space into $\mathbb{R}^2$ using a chart. The basis vectors $\frac{\partial}{\partial u_1}\Big|_p$ and $\frac{\partial}{\partial u_2}\Big|_p$ are mapped onto the new basis vectors $\left(\frac{\partial x_1}{\partial u_1}\frac{\partial}{\partial x_1} + \frac{\partial x_2}{\partial u_1}\frac{\partial}{\partial x_2}\right)\Big|_{\varphi(p)}$ and $\left(\frac{\partial x_1}{\partial u_2}\frac{\partial}{\partial x_1} + \frac{\partial x_2}{\partial u_2}\frac{\partial}{\partial x_2}\right)\Big|_{\varphi(p)}$. Thus, $\frac{\partial}{\partial x_i}$ corresponds to $\mathbf{e}_i$ in $\mathbb{R}^2$.

Applying this transformation, we obtain a new vector field $\tilde{v}(\tilde{p}) = \sum_{i=1}^{2} a_i(\varphi^{-1}(\tilde{p})) \cdot \left(\frac{\partial x_1}{\partial u_i}\frac{\partial}{\partial x_1} + \frac{\partial x_2}{\partial u_i}\frac{\partial}{\partial x_2}\right)\Big|_{\tilde{p}}$. This vector field is defined on $B \subset \mathbb{R}^2$ and maps into $\mathbb{R}^2$. As the a_i are defined on the manifold, we apply the inverse chart φ^{-1} to map from B back onto the manifold to obtain the coefficients.

A disadvantage of this approach are the distortions of the surface that are introduced by the projection. These lead to distortions of the vector field that contradict an intuitive interpretation of the linearity that we observe in the projection. Therefore, in this work, we assume the following requirement to be fulfilled: *As long as the projection of the tangent spaces is sufficiently free of distortions, we can consider $\tilde{v}$ a vector field in planar space that is equivalent to v.*

3.4 Requirements on projections

In the vicinity of a critical point in two or three-dimensional vector fields, the flow behavior is linear, i.e., the magnitude and orientation of the vector field depends linearly on the distance from the critical point. This behavior is represented by the Jacobian matrix of the critical point. We transfer this concept to manifolds by requiring that the magnitude and orientation of the vector field depends linearly on the geodesic distance from the critical point. Given a critical point on the manifold and a chart of the vicinity of the critical point into $\mathbb{R}^2$. If the chart is sufficiently distortion-free, the projected vector field can be described by the Jacobian at the critical point in $\mathbb{R}^2$. Here, absence of distortion means, that the characteristic flow properties around the critical point are preserved by the projection.

We postulate the following requirements:

1. A linear increase in the vector field magnitudes with increasing distance to the origin (measured along the surface) should be maintained in the chart (measured with Euclidean distance).
2. Geodesic lines that touch the critical point, should be preserved as straight lines in the chart.
3. In a rotational field, particles should rotate around the center at the same frequency independent of the distance to the center, i.e., the angular velocity around the critical point should be preserved.

We designed these requirements so that the typical flow behavior around critical points is preserved by the projection. The first requirement ensures that the typical linear decrease in the vector field magnitudes that can be observed when approaching a critical point, is also present in the projection. The second requirement ensures that, e.g., star sources and sinks appear as such in the projection. The third requirement ensures a correct appearance of the rotating flow around center points.

If a chart satisfies our requirements, linear flow behavior on the manifold is carried over into the projection. Vice versa, linear flow behavior in the projected vector field is also present on the manifold. Thus, linearity on general manifolds becomes computationally accessible.

4 APPLICATION TO THE SPHERE

We demonstrate the application of our definition of linear flow behavior by considering the sphere. This two-dimensional manifold has several advantages: It has a well-known embedding into $\mathbb{R}^3$ and many parameterizations and projections into $\mathbb{R}^2$ are available. Furthermore, its constant curvature allows to reliably estimate the effects of the shape of the surface on the projections.

The main task of the application is to choose a projection that is well suited with respect to the requirements stated in the previous section. Because of the curvature of the sphere, it is impossible to meet all three requirements at the same time. For example, requirement 1 and 3 are contradictory. A corollary from the theorem of Gauss-Bonnet is that the perimeter of a circle of radius r deviates from $2\pi \cdot r$ depending on the curvature of the surface. Thus, we either distort the vector magnitudes or the angular velocity [EJ07, Küh10].

Therefore, it is not possible to find a projection that meets all requirements. In the following, we state three common projections and investigate how well they preserve different flow behaviors. Finally, we introduce an approach to deal with the distortions due to the projections by using local charts.

4.1 Projections and their Properties

There exists a large variety of projections that preserve different properties in the final map [Sny97]. Especially *conformal* and *equal-area* projections are contrasted. Conformal projections locally preserve angles, respectively shapes. In contrast to this, equal-area projections try to preserve the area measures, which naturally induces distortions.

In the following, we consider the parameterization of the unit sphere in sphere coordinates (ϕ, λ) with latitude ϕ and longitude λ. A projection is a function $(\phi, \lambda) \mapsto (x, y) \in \mathbb{R}^2$ that directly maps from the parameterization into the plane. If applicable, (ϕ_0, λ_0) denotes the origin of the projection. We assume the vector field to be given in (ϕ, λ)-coordinates. I.e., for every position $(\phi_{arb}, \lambda_{arb})$ we know the velocity and direction in the tangential space $\left\{\frac{\partial}{\partial\phi}, \frac{\partial}{\partial\lambda}\right\}\Big|_{(\phi_{arb},\lambda_{arb})}$.

We define the mapped tangential spaces by computing the basis $\left\{\left(\frac{\partial x}{\partial \phi}\frac{\partial}{\partial x}+\frac{\partial y}{\partial \phi}\frac{\partial}{\partial y}\right),\left(\frac{\partial x}{\partial \lambda}\frac{\partial}{\partial x}+\frac{\partial y}{\partial \lambda}\frac{\partial}{\partial y}\right)\right\}$ for the position (x_{arb}, y_{arb}). Thus, the projection $\mathbf{v}_{proj}$ of a vector $\mathbf{v} = (v_\phi, v_\lambda)$ is given by

$$\mathbf{v}_{proj} = \begin{pmatrix} \frac{\partial x}{\partial \phi}\cdot v_\phi + \frac{\partial x}{\partial \lambda}\cdot v_\lambda \\ \frac{\partial y}{\partial \phi}\cdot v_\phi + \frac{\partial y}{\partial \lambda}\cdot v_\lambda \end{pmatrix}.$$

To study the flow behavior in projected vector fields, we use the following three projections.

Orthographic projection – The *orthographic projection* is given by

$$\begin{aligned} x &= \cos(\phi)\cdot\sin(\lambda-\lambda_0), \\ y &= \cos(\phi_0)\cdot\sin(\phi) - \sin(\phi_0)\cdot\cos(\phi)\cdot\cos(\lambda-\lambda_0). \end{aligned}$$

The corresponding basis vectors are

$$\begin{aligned} \frac{\partial x}{\partial \phi} &= -\sin(y)\cdot\sin(x-x_0), \\ \frac{\partial x}{\partial \lambda} &= \cos(y)\cdot\cos(x-x_0), \\ \frac{\partial y}{\partial \phi} &= \cos(y_0)\cdot\cos(y) \\ &\quad +\sin(y_0)\cdot\sin(y)\cdot\cos(x-x_0), \\ \frac{\partial y}{\partial \lambda} &= \sin(y_0)\cdot\cos(y)\cdot\sin(x-x_0). \end{aligned}$$

It is one of the most common and easiest projections. As shown in Figure 1(b), only the hemisphere facing the tangential plane can be projected. Trivially, it is neither a conformal nor an equal-area projection. Thus, shapes are clearly distorted in the projection and the distortion drastically increases towards the boundaries. However, angular speeds around the map's origin are preserved.

Mercator projection – The *Mercator projection* is

$$x = \lambda - \lambda_0, \qquad y = \operatorname{asinh}\big(\tan(\phi)\big).$$

The basis vectors are

$$\begin{aligned} \frac{\partial x}{\partial \phi} &= 0, & \frac{\partial y}{\partial \phi} &= \sqrt{(\cos(y))^{-2}}, \\ \frac{\partial x}{\partial \lambda} &= 1, & \frac{\partial y}{\partial \lambda} &= 0. \end{aligned}$$

It was developed by Gerhard Mercator (1512–1594). It maps the sphere to a surrounding cylinder and introduces an appropriated distortion along the cylinder axis to produce a conformal mapping (cf. Figure 1(c)). Thus, small scale geometrical shapes are well preserved. Unfortunately, this projection is not equal-area and the great circles of the sphere are not mapped to straight lines.

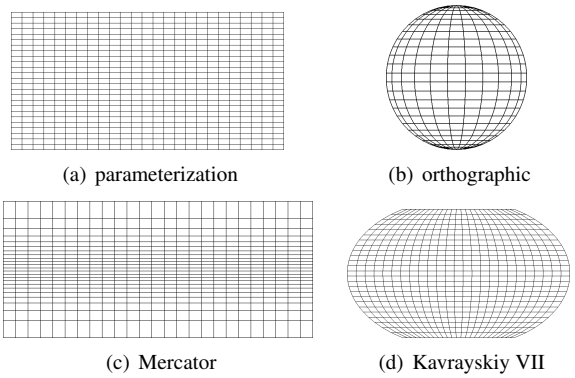
Kavrayskiy VII – The Kavrayskiy VII projection was introduced by Valdimir Vladimirovich Kavrayskiy (1884–1954). The projection is

$$x = \frac{3}{2}\cdot\lambda\cdot\sqrt{\frac{1}{3}-\left(\frac{\phi}{\pi}\right)^2}, \qquad y = \phi.$$

The basis vectors are

$$\begin{aligned} \frac{\partial x}{\partial \phi} &= \frac{-\frac{3}{2}\cdot x\cdot y}{\pi^2\cdot\sqrt{\frac{1}{3}-(\frac{y}{\pi})^2}}, & \frac{\partial y}{\partial \phi} &= 1, \\ \frac{\partial x}{\partial \lambda} &= \frac{3}{2}\cdot\sqrt{\frac{1}{3}-(\frac{y}{\pi})^2}, & \frac{\partial y}{\partial \lambda} &= 0. \end{aligned}$$

It was designed to provide a compromise between a perfect equal-area and conformal projection. The obtained map exhibits only small overall distortions (cf. Figure 1(d)). Thus, the Kavrayskiy VII projection is a good candidate for general purposes and is the last mapping we use for our study.



(a) parameterization (b) orthographic (c) Mercator (d) Kavrayskiy VII

Figure 1: Figure 1(a) shows the parameterized grid of a sphere and 1(b) to 1(d) its projected counterparts. While the orthographic projection nicely represent the sphere's shape, it unfortunately discards one hemisphere. The other two projections realize a conformal (cf. 1(c)), and a compromise between equal-area and conformal projection (cf. 1(d)).

4.2 Atlas-Based Decomposition

The distortion caused by a projection commonly increases with the distance from the map's origin. Thus, linear flow behavior on a sphere, as defined in Section 3.4, usually can not be preserved globally in a projection. This triggers the natural idea to use several charts that only project small parts of the manifold.

By using multiple charts that locally project the vector field into $\mathbb{R}^2$, we obtain an *atlas-based representation* of the vector field. The construction of the atlas has to ensure that the obtained vector field projections are almost distortion-free. For example, one could use every critical point of the sphere's vector field as the projection center for one chart to preserve the flow behavior in its vicinity as good as possible.

Furthermore, we can generalize the approach of local projections to non-critical points. Then, the charts can be based on regions on the manifold with mostly affine linear flow behavior in order to obtain a simple and easy comprehensible representations of the vector field. This idea corresponds to the work of Koch et

al. [KWKH13, KKW+15], who determined regions of mainly linear flow behavior in vector fields.

5 RESULTS

We demonstrate the influence of projections on the representation of linear flow behavior on manifolds by considering Affine Linear Neighborhoods (ALNs). Generally, an ALN is defined as

$$L_{\mathbf{x}_s} = \left\{ \mathbf{y} \in \mathbb{R}^3 \,\middle|\, \|\mathbf{v}(\mathbf{y}) - J_{\mathbf{x}_s} \cdot (\mathbf{y} - \mathbf{x}_s) - \mathbf{v}(\mathbf{x}_s)\| < E_{max} \right\}.$$

For a seed point $\mathbf{x}_s$, the ALN represents the set of positions that can be approximated by the linear vector field that is given by the Jacobian $J_{\mathbf{x}_s}$ and the offset vector $\mathbf{v}(\mathbf{x}_s)$, while staying below a specific approximation error E_{max}. In the case of vector fields on manifolds, we consider these approximations in the local charts, as described in Section 4.2. Thus, we can infer the extension of the linearly describable flow on the manifold by mapping back all positions of the ALN in the projection.

5.1 Synthetic Vector Field

To evaluate the requirements on projections of Section 3.4, we designed different synthetic datasets. These fields show different types of linear flow behavior. This allows us to study the impact of the projection on all three requirements separately.

Since the orthographic projection is very common and the effects of this projection are clearly visible, we first limit our considerations to this projection type, before introducing the results of the proposed atlas-based vector field decomposition (cf. Section 4.2).

Constant Tangential Vector Field – To investigate the influence of the projection on the flow magnitudes, we generate two constant, tangential vector fields. One field is tangent to the circles of latitude

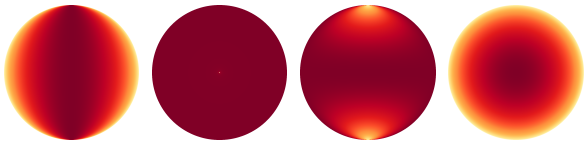
$$\mathbf{v}(\mathbf{s}) = \begin{pmatrix} \frac{1}{\cos(\lambda)} \\ 0 \end{pmatrix} \tag{1}$$

and one tangential to the lines of longitudes

$$\mathbf{v}(\mathbf{s}) = \begin{pmatrix} 0 \\ -1 \end{pmatrix}, \tag{2}$$

with $\mathbf{s} = (\phi, \lambda)$, $\phi \in [-\pi, \pi)$, and $\lambda \in [-\frac{\pi}{2}, \frac{\pi}{2}]$. The factor $\frac{1}{\cos(\lambda)}$ compensates the non-linear magnitude changes due to the decreasing radii of the circles of latitude towards the poles.

The parts of the vectors that point into the viewing direction of the projection will vanish and cause visible magnitude changes. This is clearly observable in Figures 2(a) and 2(d). If all vectors are parallel to the projection plane, the vectors will not be changed due to the projection. This is shown in Figure 2(b). Here, only the discontinuity in the pole stands out.



(a) lat. (side) (b) lat. (top) (c) long. (side) (d) long. (top)

Figure 2: These color maps show the magnitude changes of two constant, tangential vector fields due to the orthographic projection (cf. Equations 1 and 2). Figure 2(a) shows the field tangential to the circle of latitudes with the projection center in $(0,0)$ and 2(b) with the projection center $(0, \frac{\pi}{2})$. Figures 2(c) and 2(d) analogously show the field tangential to the lines of longitude. The used color map depicts vectors with a magnitude of one in dark red and zero vectors in white.

Linear Vector Field – According to requirement 1 in Section 3.4, we generate two further synthetic vector fields where the vector magnitudes increase linearly with the geodesic distance from the north pole. One field has a *center* point

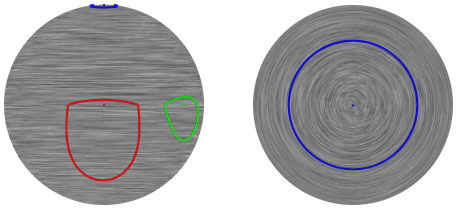
$$\mathbf{v}(\mathbf{s}) = \begin{pmatrix} (\frac{\pi}{2} - |\lambda|) \cdot \frac{1}{\cos(\lambda)} \\ 0 \end{pmatrix} \tag{3}$$

and one has a *source* in the north pole of the sphere

$$\mathbf{v}(\mathbf{s}) = \begin{pmatrix} 0 \\ -\frac{\pi}{2} - |\lambda| \end{pmatrix}. \tag{4}$$

Additionally to the factor $\frac{1}{\cos(\lambda)}$, described in the previous paragraph, the factor $(\frac{\pi}{2} - |\lambda|)$ incorporates the distance from the north pole along a fixed line of longitude. It causes a linear increase of the vector magnitudes with increasing distance from the pole.

Please note, requirement 3 can not be satisfied by this field design. The radii of circles of latitude grow slower than linear with the geodesic distance to the pole. Thus, a linear increase in vector magnitudes also leads to an increase in angular speed around the pole. The projections that we use here can not compensate this.



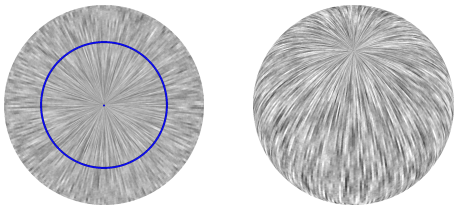
(a) $\mathbf{x}_{mc} = (0,0)$ (b) $\mathbf{x}_{mc} = (0, \frac{\pi}{2})$

Figure 3: These LIC images show the center field described by Equation 3. The depicted regions are ALNs seeded in $(0,0)$ (red), $(0.8 \cdot \pi, 0)$ (green), and $(0, \frac{\pi}{2})$ (blue). For each ALN, $E_{max} = 0.1$ was used.

Figure 3 shows two *Line Integral Convolution* [SH95] (*LIC*) images of the center point field. When the projection center is on the equator of the sphere (Figure 3(a)),

the magnitudes of the tangential vector field approach zero towards the boundary of the projection. Thus, the flow behavior gets more and more non-linear towards the boundary. This is demonstrated by the differently seeded ALNs. When the projection center is at the north pole (Figure 3(b)), the tangential vectors are parallel to the projection plane and the change in vector magnitude is only due to the lengthening along the circles of latitude. The flow directions and angular speed around the center point are preserved in this top-view.

Summarizing, we are only able to approximate small regions by a linear function. One can clearly see by Figure 3(a) that the extracted ALNs get smaller and smaller towards the projection's domain boundary. These distortion can be minimized by using a map, where the map center is also the ALN seed point (cf. Figure 3(b)).



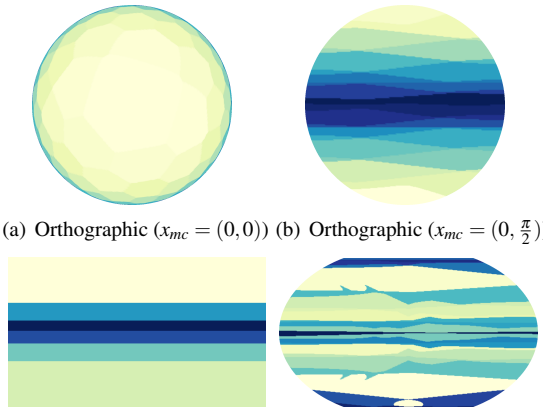
(a) $\mathbf{x}_{mc} = (0, \frac{\pi}{2})$ (b) $\mathbf{x}_{mc} = (0,1)$

Figure 4: Analogously to Figure 3(b), the left image shows the source field of Equation 4 with an ALN ($E_{max} = 0.1$) seeded in $(0, \frac{\pi}{2})$. The right image shows the same field projected with $(0,1)$ as the map center.

Figure 4 shows LIC images of the second synthetic vector field, the source field. Analogously to Figure 3(a), the projected vector field approaches zero towards the boundary. Again, the non-linear magnitude change due to the projection and the scaling of the field along the lines of longitude have the effect that the projected field can not be completely described by a affine linear field in the projection. But if the critical point is located directly in the map center, all geodesic curves that touch it are mapped to straight lines. Thus, requirement 2 is satisfied. Figure 3(b) shows the corresponding counterexample when the critical point is further away from the map center. Here the non-linearity of the projection causes a qualitatively different course of the streamline.

To summarize our first findings, only a region around the map center is sufficiently distortion-free, so that realistic analysis results can be obtained. Thus, we think that it is necessary to use an approach like the proposed atlas-based vector field decomposition for the analysis of linear flow behavior on spheres.

For the comparison of different projection types, we considered the source field (Equation 4). The projected vector fields were segmented into regions that can be linearly approximated without exceeding a specific approximation error. For this, we used the segmentation



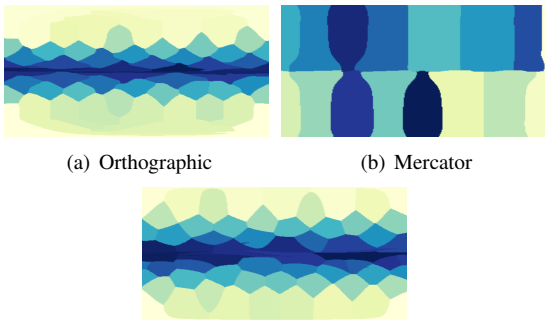
(a) Orthographic ($x_{mc} = (0,0)$) (b) Orthographic ($x_{mc} = (0, \frac{\pi}{2})$)

(c) Mercator (d) Kavrayskiy VII

Figure 5: Segmentation using the algorithm of Koch et al. [KKW+15]. The color map shows the seeding order of the segments from white to blue. Figure 5(a) shows 127, 5(b) 20, 5(c) 6, and 5(d) 24 segments.

algorithm of Koch et al. [KKW+15]. Their method decomposes a vector field into regions that are at least ALN subsets. To allow the extraction of the best and largest regions with a linear flow behavior, a heuristic based on the field's derivatives is used to predict the needed ALN seed points. The ALN-computation is iteratively applied until all positions in the field are assigned to at least one ALN. If the assignment of a point to the ALNs is not unambiguous during this iterative process, it gets assigned to that ALN, which approximates the corresponding vector value best.

The resulting segmentations reveal which parts of the projections can be used for a realistic representation of the vector field, similar to the ALNs that we discussed above. Figure 5 shows the segmentation results for the different projection types. The orthographic projection generates the largest linearly approximable region around the critical point, when the critical point is also the map center (Figure 5(a)). However, because of the distortions towards the projection boundary, this projection also generates the most segmentation regions. When the map center is at the equator (Figure 5(b)), a moderate number of segments is created. The best approximations can be achieved near the poles and the sizes of the segments slightly decrease towards the map center here. The Mercator projection (Figure 5(c)) produces the fewest segments and very large regions around the poles that can be well linearly approximated. This is a result from the conformality of this projection, since the preservation of angles has a great impact on the distortion of the vector magnitudes here. The Kavrayskiy VII projection (Figure 5(d)) is not conformal and leads to more and smaller segments, i.e. a smaller part of the projection has a meaningful interpretation. Still, fewer segments are created than in the case of the orthographic projection.



(a) Orthographic (b) Mercator

(c) Kavrayskiy VII

Figure 6: Atlas-based vector field decomposition of the source field in the (ϕ,λ)-parameter space. Each region can be well linearly approximated and corresponds to one chart of the atlas. Figure 6(a) shows 55, 6(b) 16, 6(c) 61 regions. The small number of regions can be obtained due to the absence of high distortions that occur in the different projections.

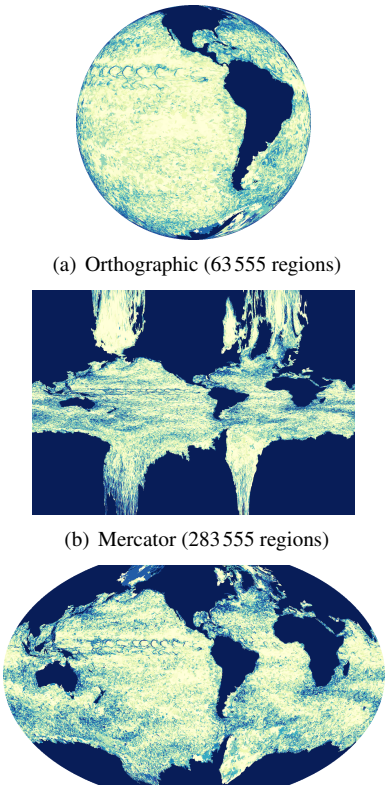
In sum, the Mercator projection appears to represent the linear vector fields best in this synthetic example. The orthographic projection is worst, depending whether the map center is aligned with the critical point or not.

Atlas-based Vector Field Decomposition – To demonstrate the construction of an atlas-based vector field decomposition, we again use the segmentation algorithm of Koch et al. [KKW+15] and adapt it to the sphere. For every seed point candidate, the vector field is projected into $\mathbb{R}^2$ while keeping the seed point at the map center. Thus, the heuristic of the original algorithm can be applied to the projected field to predict the region with the most linear flow behavior. For the found seed point, an ALN is computed in the projection to determine the extension of the new chart of the desired atlas. This process is applied iteratively until the complete field is segmented. Because the original algorithm was designed to compute the largest possible regions, the number of charts of the atlas should be minimal.

On the example of the source field (Equation 4), we compare the results for different projection types. Figure 6 shows the results. There is only a small difference between the Kavrayskiy VII projection and the orthographic projection. In both cases, many charts are needed, especially near the equator. The Mercator projection results in the simplest atlas, consisting of only 16 charts. This shows again, that the conformality, i.e., the preservation of angles in the projection, is very beneficial for the analysis of this synthetic example.

5.2 Real-World Example

To provide a real-world example, we consider the ocean currents of the whole world in six meter depth. This dataset consists of 8 618 400 grid positions.



(a) Orthographic (63 555 regions)

(b) Mercator (283 555 regions)

(c) Kavrayskiy VII (180 978 regions)

Figure 7: Segmentation of Koch et al. [KKW+15] ($E_{max} = 10^{-4}$), applied to projections of the ocean currents. It shows a high number of approximated regions and high distortions of the shapes of the continents.

When looking at the naïve segmentation of this dataset in Figure 7, we see the same shortcomings as in the synthetic fields. For example, the orthographic projection again produces a very high number of segments, due to the high non-linearity towards the domain boundary. The high distortions of shapes on the sphere are also visible in the projections (cf. Australia in Figure 7(b)). Furthermore, from the previous section, we know that the linear regions in these projections are non-realistic.

The results of the atlas-based vector field decomposition are very similar. That is why we only show the example of the orthographic projection in Figure 8, Nevertheless, we know that the Kavrayskiy VII already introduces vector field distortions at the map center. That is why, we do not recommend this projection for our approach. In contrast to the first results, both, the orthographic, as well as the Mercator projection, provide good results here. Since the flow is very turbulent, the linearly approximable charts are mostly small in size and therefore satisfy our requirements on projected linear flow behavior very well in vicinity of the corresponding map centers.

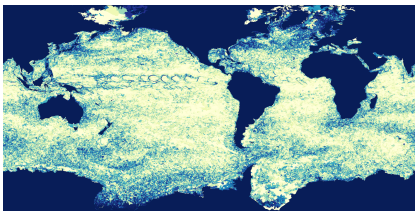


Figure 8: Atlas of the ocean currents using an orthographic projection for the vector field decomposition. The atlas contains 204 659 charts. Because of the complex structures in the flow, the charts are relatively small and therefore sufficiently distortion free.

6 CONCLUSION AND FUTURE WORK

Characterizing linearity on general 2D manifolds is a difficult problem, because tangent vectors at different points of the manifold can not directly be compared against each other. We propose a local representation of vector fields in $\mathbb{R}^2$ using projections. For this, we unify the tangent spaces at different points on the manifold by projecting them into a common space, in this case into the $\mathbb{R}^2$. Our proposed requirements on the projection ensure the preservation of certain properties of the original vector fields that are important for characterizing linear flow behavior. The projected field is accessible to known analysis methods and can be investigated as a substitute of the original field on the manifold.

The concept is evaluated on the example of the sphere. We compare different projection types regarding their suitability according to the requirements. A decomposition of the vector field into multiple charts, that are each sufficiently free of distortions and together represent the whole vector field, is proposed and demonstrated for both, synthetic examples of simple linear vector fields, as well as a real world example of ocean currents.

This is only a first step to investigate linearity on 2D manifolds. Future work can improve the way the tangent spaces are unified, evaluate the influence of changes in curvature on the quality of the projection, and also construct a design space of projection techniques that allows to choose the best possible projection for the analysis task at hand.

7 ACKNOWLEDGMENTS

Special thanks go to the *FAnToM* development group for providing the visualization software and the working environment. Furthermore, we thank James Ahrens for the access to the climate datasets via ESG-NCAR. The first two authors contributed equally to this work.

8 REFERENCES

[AH11] Auer, C. and Hotz, I., Complete tensor field topology on 2d triangulated manifolds embedded in 3d, In CGF, vol. 30, pg. 831–840, 2011.

[BHSH14] Bujack, R., Hotz, I., Scheuermann, G., and Hitzer, E., Moment invariants for 2d flow fields using normalization, In IEEE PacificVis 2014, pg. 41–48, 2014.

[EG79] Eisenberg, M. and Guy, R., A proof of the hairy ball theorem, Am. Mathematical Monthly, pg. 571–574, 1979.

[EJ07] Eschenburg, J.-H. and Jost, J., Differentialgeometrie und Minimalflächen, Springer, 2007.

[FW09] Fuselier, E.-J. and Wright, G., Stability and error estimates for vector field interpolation and decomposition on the sphere with rbfs, SIAM Journal on Numerical Analysis, vol. 47(5), pg. 3213–3239, 2009.

[Hal01] Haller, G., Lagrangian structures and the rate of strain in a partition of two-dimensional turbulence, Physics of Fluids (1994-present), vol. 13(11), pg. 3365–3385, 2001.

[HSD04] Hirsch, M.-W., Smale, S., and Devaney, R.-L., Differential Equations, Dynamical Systems, and an Introduction to Chaos, vol. 60, Academic Press, 2004.

[KKW+15] Koch, S., Kasten, J., Wiebel, A., Scheuermann, G., and Hlawitschka, M., 2d vector field approximation using linear neighborhoods, The Visual Computer, 2015.

[KWKH13] Koch, S., Wiebel, A., Kasten, J., and Hlawitschka, M., Visualizing linear neighborhoods in non-linear vector fields, In IEEE PacificVis 2013, pg. 249–256, 2013.

[Küh10] Kühnel, W., Differentialgeometrie, Springer, 2010.

[LP06] Llibre, J. and Pessoa, C., Homogeneous polynomial vector fields of degree 2 on the 2-dimensional sphere, Extracta Mathematicae, vol. 21(2), pg. 167–190, 2006.

[SRWS10] Schneider, D., Reich, W., Wiebel, A., and Scheuermann, G., Topology aware stream surfaces, CGF, vol. 29(3), pg. 1153–1161, 2010.

[Sny97] Snyder, J.-P., Flattening the earth: two thousand years of map projections, University of Chicago Press, 1997.

[SH95] Stalling, D. and Hege, H.-C., Fast and resolution-independent line integral convolution, In SIGGRAPH '95, pg. 249–256, 1995.

[WKS12] Wiebel, A., Koch, S., and Scheuermann, G., Glyphs for non-linear vector field singularities, In TopoInVis Workshop 2011, pg. 177–190. Springer, 2012.

[ZMT06] Zhang, E., Mischaikow, K., and Turk, G., Vector field design on surfaces, ACM Transactions on Graphics, vol. 25(4):, pg. 1294–1326, 2006.

ISSN 2464-4617 (print) WSCG 2016 - 24th Conference on Computer Graphics, Visualization and Computer Vision 2016
ISSN 2464-4625 (CD-ROM)

Accelerating Spatial Data Structures in Ray Tracing through Precomputed Line Space Visibility

Kevin Keul
University of Koblenz-Landau, Koblenz, Germany
keul@uni-koblenz.de

Stefan Müller
University of Koblenz-Landau, Koblenz, Germany
stefanm@uni-koblenz.de

Paul Lemke
University of Koblenz-Landau, Koblenz, Germany
lemke@uni-koblenz.de

ABSTRACT

We propose an efficient approach to precompute and reuse visibility information based on existing spatial data structures by using a precomputed data structure: the line space. This data structure provides an additional skip condition by checking whether the subnodes in a hierarchical spatial data structures need to check for intersection with the ray. We evaluate this method on different test scenes and show that it is able to achieve a remarkable speed-up by using this skip condition. Furthermore we describe algorithms for fast set-up and traversal in detail and discuss important strategies for this approach.

Keywords

Visualization, Computer Graphics, Ray Tracing, Data Structures, Visibility Algorithms

1 INTRODUCTION

The basic principle of ray tracing is that every visual effect is computed with rays that search for the nearest primitive in a given direction from a known starting point. When this intersection is found, more rays starting from there on can be processed. With this it is easy to calculate effects like shadows, reflexions and refractions with only one additional ray for each effect. With even more additional rays one can compute complex visual effects such as ambient occlusion or indirect lighting.

However, the quality of rendering comes with long rendering times, where even the slightest improvement can make a significant difference. The main limiting factor is the time it needs to compute the nearest intersection with the scene geometry. Therefore it is important to use an acceleration data structure which supports the task of finding the nearest intersection in an efficient way. Many of the data structures used today aim to subdivide the scene or the world space in such a way that the scene is equally distributed over every subunit in the data structure.

While this is already a studied field of research our approach goes beyond that. We try to precompute visibility tests on possible directional shafts additionally to the main data structure. Those precomputations should support the data structure by providing it with an additional condition to decide if it is possible to skip the main intersection computations. With this we achieve a performance speed-up compared to the already created acceleration data structure. Though it is a directional precomputation, we compute every possible direction and so enable visibility tests on the whole space with every possible starting and end point. Through this it is not only a speed-up for the initial coherent rays, but for every possible ray.

Moreover we try to combine the ideas of existing spatial data structures and extend the used traversal algorithms to optimize the achieved speed-up. For this we build a tree with a higher branching factor compared to the typically used data structures like the octree. In this paper we call it the N-tree because of the arbitrarily branching factor which can be dynamically chosen. With a higher factor it is possible to skip more spatial groups of elements thanks to a single test with our directional data structure.

2 RELATED WORK

In the past decades numerous data structures for accelerating ray tracing have been created and improved. Most of them aim to reduce intersection computations with the scene geometry by using spatial subdivision of the scene itself. An obvious way for this is to divide the total space with a simple cartesian grid, called the uniform grid, where every cell, called voxel, has the same size. For the traversal of this data structure it is possible to use known algorithms which are mostly based

on Bresenham's 2D line drawing algorithm [Bre65], for example the 3D-DDA (3D Digital Differential Analyzer) algorithm introduced by Amanatides and Woo [AW87]. Today the use of grids benefits from quite efficient voxelization algorithms [ED06][ED08][SS10]. The biggest disadvantage of uniform grids is the variance between cells, so that in most cases there not only exist cells containing many scene candidates but also cells which are completely empty. Nevertheless it was shown that in some cases the use of uniform grids results in a significant performance gain in ray tracing [HKH11].

Hierarchical data structures are one kind of improvement. The goal is to have a high level of hierarchy and therefore a high resolution in those areas where there are many scene objects. Recursive grids were shown to work well with objects of varying density by recursively subdividing those voxels of the grid containing many scene candidates. Jevans and Wyvill [JW88] used an adaptive subdivision method where the branching factor of a voxel was higher the more scene candidates it contains. Octrees have a constant branching factor of 8 subvoxels per voxel. All subvoxels within a voxel have the same size so the subdivision of a voxel is exactly in the center point. The traversal can be done bottom-up, as by Samet [Sam89], or top-down, as introduced by Revelles, Ureña and Lastra [RUnL00]. By using KD-trees [Hav00] one tries to achieve better distributions of scene objects to subvoxels as octrees. For this the split of the voxel is not necessarily in the center point but along the axis aligned plane, which seperates the containing scene objects in half. Extensions try to improve the scalability via SIMD commands [WPS*03][RSH05] and GPU advantages in stack based implementations [EVG04][FS05] as well as stackless implementations [PGSS07]. Binary space partitioning (BSP) trees, subdividing the space along arbitrary axes, have been used [SS92][KM07] and as shown in [TI08] they are more efficient as KD-Trees but need longer build times due to more complex construction algorithms.

Another approach to reduce computational overhead is to use bounding volumes around scene objects instead of spatial ordering. Bounding volume hierarchies (BVH) [KK86] apply k-DOPs, spheres or other kinds of proxy geometry and the traversal applies typical tree search and sorting techniques to reduce the complexity. Bounding interval hierarchy (BIH), a variation of axis-aligned bounding box trees, was used to great extent [NS04][WK06]. Advancements of these try to use SIMD parallelism [RSH05]. One way to do this is to use multi bounding volume hierarchies (MBVH) which in contrast to regular BVH store an arbitrarily number of subnodes according to the level of SIMD instructions [EG08]. Recently there have been implementations for the GPU using stackless MBVH [ÁSK14] and GPU accelerated construction [KA13].

Other acceleration methods try to take the visibility into account. Arvo and Kirk presented 5D volume structures, starting at a 3D object with a 2D angle [AK87]. They achieved a notable performance gain but due to its camera dependence it needs to be rebuild regularly whereas our data structure is independent from the camera position. Visibility preprocessing for urban scenes was used in the way of identifying blocker primitives by Bittner et al. [BWW01] and Leyvand et al. [LSCO03]. They also use the notation "line space" but it has a different meaning compared to our usage. Visibility precomputations have been a big topic in radiosity calculations [CW12]. In this context Drettakis and Sillion [DS97] used line space computations to precompute visibilities in a very similar way as we do. In their paper a line is considered as a link between two arbitrary surface elements surrounded by a shaft, covering all potential rays between both surface elements. Shaft culling was further used to optimize radiosity calculations by Haines and Wallace [HW94].

3 OVERVIEW

Our goal is to extend typical hierarchical acceleration data structures by precomputated visibility tests based on lines and shafts. With this the extended data structure performs just one additional visibility operation per node traversal for a given ray, which is done right before the intersection tests of the ray with the subjects within the current node. If this operation fails, the following intersection tests of the ray and the node subjects can be skipped completely. Note that the subjects of the node can be the objects of the scene contained by this node as well as all its own subnodes. Like most acceleration data structures we do not aim to work with dynamic scenes, so the set-up of the data structure does not need to be able to compute in real time. Our goal is to speed up ray tracing of static scenes and therefore only compute the data structure once initially.

3.1 *N*-tree as initial data structure

As the base data structure we use the *N*-tree, a variation of the recursive grid [JW88] with fixed branching factor, which benefits the most from our visibility data structure, due to reasons which are explained later on. Every edge of one *N*-tree node is divided in *N* equally long parts. We need to have our subnodes equal in size for our visibility test, which is also explained later. Therefore, we are not able to use arbitrary splitting points like in KD-Trees, where different subnodes of one node may differ in size. Although it is possible to store scene objects (the candidates) in every hierarchical level of the *N*-tree, our performance results suggest that only leaf nodes should contain candidates. Every

node of the N-tree is either a leaf node and contains scene objects as candidates or consists of $N \times N \times N$ subnodes.

One can easily observe that the two main variables, N and the maximum depth of the tree (for further examples d), can be arbitrarily chosen and different selections of the values can give similar results. For example, do either $N = 2, d = 6$ (which resembles the typical octree) as well as $N = 8, d = 2$ result in a resolution of $64 \times 64 \times 64$ entries on the deepest hierarchy level. One observable difference lies in memory consumption in sparsely filled trees, where a higher N results in more memory usage due to a higher number of empty subnodes.

Algorithm 1 The traversal algorithm

```
1:  procedure TRAVERSENODE(Ray r, Node n)
2:      p ← 0
3:      if n has primitives then
4:          p ← nearest primitive intersecting r within n
5:      else if n has subnodes then
6:          while p = 0 and subnodes left do
7:              s ← next subnode in direction of r
8:              if s is non-empty then
9:                  p ← TRAVERSENODE(r, s)
10:             end if
11:         end while
12:     end if
13:     return p
14: end procedure
```

The pseudo code of the traversal algorithm for the N-tree is shown in Algorithm 1. In principle it is divided into two parts. At first, the exact start node has to be found. Starting at the root node the next inner subnode is chosen until the leaf node is reached. With this the main traversal starts. Every processed node has either candidates, which are tested for intersection with the current ray (lines 3 and 4), or has subnodes, which are recursively processed. All candidates within a leaf node have to be tested, but if at least one intersection is found, the traversal algorithm can stop. The step from one node to the subnodes follows a top-down strategy as proposed by Revelles et al. [RUnL00]. Like explained above, in our case it is not possible that a node has both, candidates and subnodes. As proposed by Amanatides et al. [AW87], the subnodes are traversed in a grid like manner (line 7). If a subnode neither contains candidates nor subnodes, it does not need to be processed at all and can be skipped in the traversal (lines 8-10). The algorithm continues with the next subnode. In the following, those subnodes are called ”empty”. The loop can stop if a primitive is found within a subnode (lines 6).

Figure 4a shows an exemplary traversal of the N-tree. The ray starts at the origin O within the node starting from S. At this point, every intersected subnode needs to be checked although neither the geometry nor any subnode containing the geometry is intersected by the ray.

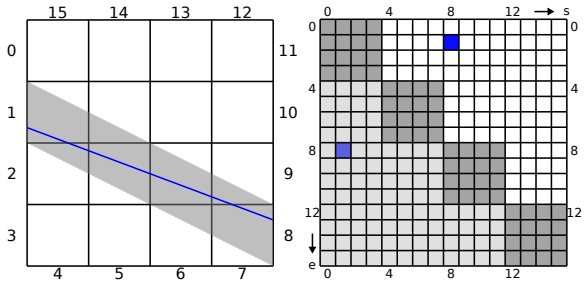
3.2 Visibility Information with the line space

The line space builds upon the presented N-tree and extends it with an additional visibility test which decides whether a node can be skipped in the traversal. Note that this additional skip condition still works if the node has both, candidates and subnodes. Like explained above, a node contains of $N \times N \times N$ subnodes. Furthermore, each side of the nodes' bounding volume divides in $N \times N$ smaller sides with equal size, which makes a total of $6 \times N \times N$ smaller sides in the volume. These smaller sides are countable and each of these gets its own identifiable index. It is now possible to create shafts from every possible index to every other possible index. For each of those shafts it is decidable whether there exists at least one subnode partially or in total within the shaft that contains either candidates or subnodes itself. If a shaft has only empty subnodes, in other words the shaft does not intersect any subnode that is non-empty, the shaft itself is called empty.

The line space for a given node contains the information whether a shaft is empty or non-empty for every possible shaft within this node. It can be represented as 2D array or texture where the first axis stands for every possible start index and the second axis stands for every possible end index of sides. So, the pixel with the coordinates x and y denotes the shaft starting at the side with the index x and ending at the side with the index y. The value of the pixel represents whether the corresponding shaft is empty or intersects with at least one non-empty subnode.

In the step of deciding whether a shaft is empty, we use subnodes instead of the discrete scene geometry for two reasons. On the one hand, the scene geometry is already arranged in the subnodes of the N-tree and possibly quite many primitives of the scene result in just a few subnodes. On the other hand, the correspondence between the shafts and all intersected subnodes can be precomputed resulting in masking, which is further explained in the next paragraph. For these reasons it is possible to accelerate the construction of the line space effectively. One drawback to this is that there might be some subnodes within a shaft that only contain primitives of the scene that do not intersect with the shaft. Therefore, it would not be necessary to mark the corresponding entry in the line space. Anyway, it is marked because of the intersection between the subnode and the shaft. This results in possibly longer calculation times

during traversal but especially with a big N it becomes negligible.



(a) (b)

Figure 1: (a) An empty 2D scene with one exemplary ray and the belonging shaft between the start index 1 and the end index 8 and (b) the corresponding line space where the shaft is marked in blue.

Figure 1 demonstrates the relation between a node containing subnodes and the corresponding line space in 2D. There, the bounding box of a scene is subdivided into a 2D N-tree with $N = 4$ consisting of $4 \times N = 16$ elements. The border edges are numbered from 0 to 15. Each shaft is identified by the tuple of the start index and the end index of the sides. As a result the line space is of size 16×16, where each index tuple represents a shaft in the scene. The blue line in the left image is represented by index (1,8) or (8,1) respectively.

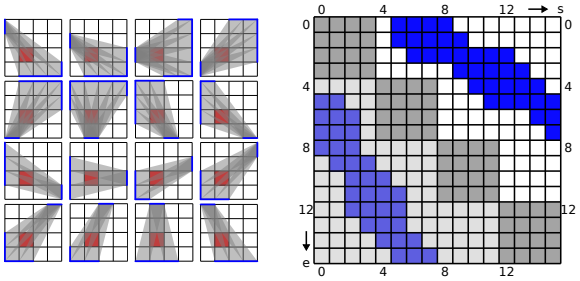
A few trivial properties help to reduce the memory capacity of the line space:

1. *LS(s; e) = LS(e; s)*: The line space is symmetric and the upper right triangle grants sufficient information.
2. *LS(s; s) = 0*: The elements of the diagonal characterize degenerated shafts with zero volume and can therefore be omitted.
3. *Coplanarity (Collinearity in 2D case)*: Shafts between coplanar sides are also degenerated, leading to blocks around the diagonal.

In 2D each of the 4 bounding sides contains N subsides so the total number of entries in the line space is $4N \times 4N = 16N^2$. Using the collinearity this can be reduced by $4N^2$ and afterwards divided in half due to symmetry, resulting in a total number of entries of size $6N^2$. In 3D each of the 6 bounding sides of the bounding box of the node contains $N \times N$ subsides and therefore the line space has $6N^2 \times 6N^2 = 36N^4$ entries in total. In the same way as for the 2D case this can be reduced to a total of $15N^4$ entries due to coplanaraty and symmetry. Note that this is only the size of the line space for a single node and therefore the memory consumption is quite high with a big N. In our test cases we found that $N = 10$ is sufficient for most cases and the memory consumption is appropriate. All entries for one line space are stored in a list and accessed with an identifier. This identifier is independent from symmetry and results in the same entry for both of the symmetric cases.

3.3 Set-up of the line space

Figure 2 shows the relevance of one non-empty subnode (marked in red) to the line space. On the left side for each possible start index it is shown which shafts count as non-empty because of the marked subnode. The right side shows the corresponding line space where exactly those pixels are marked that belong to non-empty shafts. Note that if only the marked subnode is non-empty, the line space would always result in this outcome. It is not relevant how many scene primitives are contained in this subnode or how they are located. So, the resulting line space which is presented can serve as a mask for this subnode.



(a) (b)

Figure 2: (a) All shafts covering one subnode (red) in 2D and (b) the resulting line space (bit mask). Every shaft with start index x and end index y fills the corresponding pixel in the line space. Symmetry and collinearity of the line space are quite obvious.

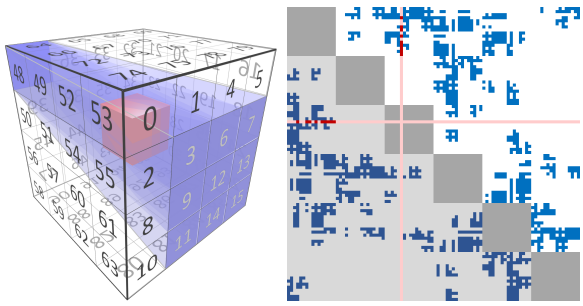
With this it is possible to precompute the masks for every possible subnode within a node and combine them to a mask atlas. This results in N^3 masks (one for each possible subnode) for the mask atlas, which then contains $N^3 \times 15N^4$ entries in total.

The pseudo code for the set-up algorithm for all line spaces of every node in the N-tree is shown in Algorithm 2. Our approach works in a top-down way starting with the root node. A line space for a node is only necessary, if the node itself contains subnodes. Every line space is computed with the help of the mask atlas. For every non-empty subnode of a node all entries of the corresponding masks are combined and result in the line space of the current node (lines 4-6). In the binary case, where it only matters whether a shaft is empty or not, this combination can be done with a simple ”or” operation for every entry of the mask with the corresponding entry in the line space. The line spaces of every subnode consisting of subnodes itself are then computed recursively (lines 7-9).

Algorithm 2 Calculation of Line Space starting in the root node

```
1:  procedure CALCLINESPACE(Node n)
2:      LS ← create LineSpace for n
3:      for all subnodes s ∈ n do
4:          if s is non-empty then
5:              mask ← mask denoted by s in n
6:              ADDMASKTOLINESPACE(LS, mask)
7:              if s has subnodes then
8:                  CALCLINESPACE(child)
9:              end if
10:         end if
11:     end for
12: end procedure
```

Figure 3 presents an example for a 3D line space. As with the previous examples, N is set to 4. It is obvious that the line space is much more complex compared to a 2D line space. Where in the 2D case every side is subdivided in 4 smaller parts, making it a total of 16 subsides, in the 3D case every of the 6 bounding sides is subdivided in 16 smaller sides and therefore making a total of 96 possible start and end sides. Figure 3b shows the mask for one subnode (marked in red). For every start index from $s \in 0..95$, a one bit entry provides the information, whether the shaft to end index $e \in 0..95$ intersects this subnode. In the shown example, we have 9 resulting shafts for the starting patch $s = 37$ which can be seen in the red column of the line space.

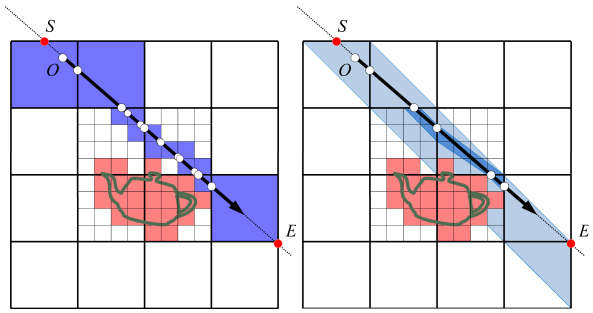


(a) (b)

Figure 3: (a) All shafts in 3D intersecting the red subnode from the start index with index 37. (b) Line space bit mask (4^3 subnodes with 962 LS-entries) for the red subnode. Note that the subnode itself can be subdivided as well and can therefore include its own line space.

3.4 Traversal of the line space

The traversal of the line space is mostly equivalent to the traversal presented in algorithm 1. Indeed the presented algorithm is just extended by another skip condition, which can be added before the subnodes are processed (after line 5 in algorithm 1). The skip condition checks, whether the line space entry corresponding to the current node is marked. If this is not the case, it means that all subnodes within the current shaft are empty and therefore no subnode needs to be processed with the current ray. The shaft itself is determined by the precise start and end index within the node which are intersected by the ray. These have to be computed first in order to identify the shaft the current ray belongs to.



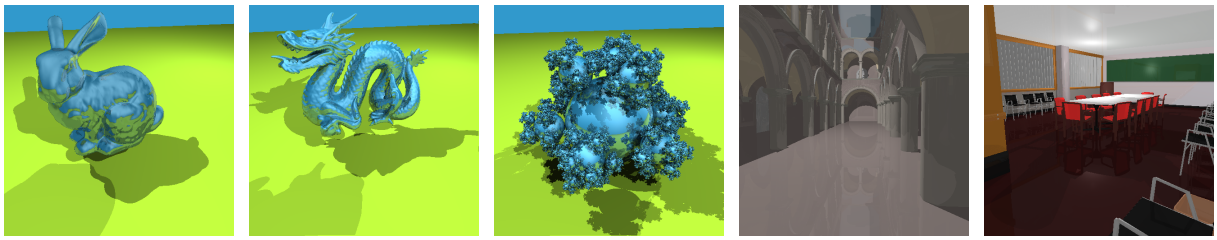
(a) (b)

Figure 4: (a) Traversal of the N-tree. Although no subnode containing geometry (red) is intersected by the ray, the algorithm traverses every possible subnode (dark blue) intersected by the ray. (b) traversal of the line space. Instead of testing every subnode intersected by the ray, it is first checked if the corresponding shaft intersects any non-empty subnodes. If this is not the case (like shown with the darker blue shafts), all subnodes within the shaft are skipped.

Figure 4 presents an exemplary traversal using the line space. For a given ray, we compute the intersection with the root node to determine the initial start index S and end index E. The x-, y-, z-coordinates of S and E are mapped to side indices of the root node surface, yielding the indices for the top level line space. In the example the top level shaft contains non-empty subnodes. Therefore, we select the subnode covering the ray origin O and from there on we start the traversal of the subnodes similar to the traversal of the N-tree. If one of these inner nodes is not subdivided, we check the candidate list of this node (if any) for intersection and continue with the next inner node, if no intersection is found. If the node is subdivided, we check the next level line space first with new start and end indices. If the shaft is not empty, we proceed with the traversal with smaller increments. In the example all inner shafts (in dark blue) corresponding to subnodes indicate that there are no non-empty inner subnodes and therefore these inner subnodes can be skipped at all.

4 RESULTS AND DISCUSSION

Our method was implemented in C++, exploiting SIMD operations (SSE) and multi-threading on a CPU. The results were evaluated on a PC with AMD Phenom II X6 1090T (6 cores, 3.5GHz) and 16 GB DDR3 RAM.



(a) BUNNY (69k triangles) (b) DRAGON (871k triangles) (c) SPHEREFLAKE (597k spheres) (d) DUBROVNIK SPONZA (66k triangles) (e) CONFERENCE ROOM (331k triangles)

Figure 5: Test scenes used for the performance measurements. Those include individual objects with a varying number of triangles (Bunny and Dragon), a fractal scene using spheres instead of triangles and architectural scenes with different number of triangles (Dubrovnik sponza and Conference room). The images were rendered using 3 light sources and multiple levels of reflection.

The used ray tracer computes intersection points for primary rays and up to 10 levels of reflections, where every primitive of the scene geometry is reflecting the ray. For every intersection with scene geometry 3 light sources are used for lighting and for each of those one shadow ray is evaluated. By using reflections and shadow rays we mostly work on more or less incoherent rays, which are traced by our method with no difference in comparison to coherent rays. All scenes were rendered with a resolution of 512×512 using different camera angles. The result is the average run time.

Multiple well-known test scenes with different characteristics and of different size of primitives have been used for evaluation (shown in figure 5). We divide those scenes in scenes showing individual objects only (Bunny and Dragon), architectural scenes (Dubrovnik sponza and Conference room) and a fractal scene (sphere flake using spheres instead of triangles). The individual objects represent the quality of the data structure for a single object only, where many primitives are concentrated in small space. For this purpose the Bunny is a model with a rather low number of primitives, whereas the Dragon consists of a lot of primitives. The architectural scenes represent conventional scenes, which may for example be used in games or films. We use the sponza as a scene with few primitives and the Conference room as scene with quite many primitives. The sphere flake is a fractal scene, which consists of a lot of primitives (spheres in our example). Those primitives are not concentrated in the center of the object, but are equally distributed.

For the N-tree and the line space we evaluate the size of the data structure and the runtime performance within our ray tracer. We compare those with the standard implementations of the uniform grid and the octree to show that the use of visibility information is an improvement of typical well-known spatial data structures. Furthermore we vary in the values of the two paremeters of the N-tree and the line space, which are the branching factor N and the maximal depth d, and investigate the differences in size and performance.

The results of the tests are shown in table 1. We evaluated several parameter sets for all data structures and only the best for each scene were considered. Note that the value of d belongs to the maximal depth of the data structure, which is not always needed. In scenes with a small number of primitives it is therefore possible that a big value of d does not provide any benefit.

The uniform grid grants good performance, especially in rather small scenes. The memory size used is in all test cases among the smallest. The optimal resolution for the uniform grid in most test scenes is 128^3 voxels in total. A higher resolution results in a higher traversal cost and a much higher memory consumption of the grid structure and might therefore only be beneficial in big scenes (like the dragon). In comparison to the uniform grid, the octree has a better performance in those big scenes (dragon and conference room), but worse in small scenes. The memory consumption depends on the value of d, where a small value results in a smaller memory consumption. In big scenes a big value of d is beneficial for performance but unfavorable for the required memory size.

The N-tree has a better performance than the octree, due to the higher branching factor, where every node is traversed in a grid-like manner. In most cases the N-tree performs similar to or better than the uniform grid, especially in the architectural scenes or in scenes with a high number of primitives. While a high value of N grants better performance, the higher branching factor results also in a bigger memory consumption, especially in sparsely filled N-trees. If a node is subdivided, it results in quite a lot of subnodes (N^3), even if only a few of them are actually needed. The optimal parameters of the N-tree in respect to the performance have been achieved with a value of N between 6 and 10. The optimal value of d is mostly either 3 or 4.

The line space, as an extension to the N-tree, is beneficial in all cases. Mostly it achieves a performance gain of up to 30% in comparison to the N-tree. In all test scenes the optimal parameters were the same as for the conventional N-tree. Obviously the additional usage

Scene		Uniform Grid	Octree	N-tree	Line Space
	parameters	128^3	d $\rightarrow 7$	$N \rightarrow 9, d \rightarrow 3$	$N \rightarrow 9, d \rightarrow 3$
BUNNY	time per frame (s)	0,111	0,137	0,123	0,101
(69k triangles)	memory (MB)	78,4	55,2	82,5	106,7
	parameters	256^3	d $\rightarrow 9$	$N \rightarrow 7, d \rightarrow 4$	$N \rightarrow 7, d \rightarrow 4$
DRAGON	time per frame (s)	0,327	0,332	0,297	0,253
(871k triangles)	memory (MB)	441,0	438,1	823,2	929,6
	parameters	128^3	d $\rightarrow 7$	$N \rightarrow 8, d \rightarrow 3$	$N \rightarrow 8, d \rightarrow 3$
SPHEREFLAKE	time per frame (s)	0,151	0,208	0,179	0,129
(597k spheres)	memory (MB)	200,8	187,9	511,6	644,0
	parameters	128^3	d $\rightarrow 10$	$N \rightarrow 10, d \rightarrow 3$	$N \rightarrow 10, d \rightarrow 3$
SPONZA	time per frame (s)	1,224	1,771	1,414	1,192
(66k triangles)	memory (MB)	80,4	55,1	220,0	294,7
	parameters	128^3	d $\rightarrow 10$	$N \rightarrow 10, d \rightarrow 3$	$N \rightarrow 10, d \rightarrow 3$
CONFERENCE	time per frame (s)	1,395	1,593	1,300	1,089
(331k triangles)	memory (MB)	213,3	190,8	236,8	249,9

Table 1: Performance evaluations for the test scenes shown in figure 5. All scenes were rendered using 3 light sources and up to 10 reflections. We have compared typical data structures (uniform grid and octree) with the N-tree without and with the usage of the line space. Only the best parameter set in terms of traversal time for each data structure and each scene is shown.

of the line space results in a bigger memory consumption, where a high value of N is especially bad, because of the high number of possible shafts ($15N^4$) for every subdivided node. Due to the fact that only non-leaf nodes need a line space, this increment in memory size is quite acceptable in comparison to the total required memory size. While high values of N and d are a disadvantage in terms of memory consumption, they can be beneficial for traversal performance. A big value of N leads to long but slim shafts referring to many but small subnodes. If the shaft is empty, it therefore allows for a quick skip of many subnodes in just one computation. Moreover, long and slim shafts contain small subnodes. Even if these subnodes are intersecting the shaft only for a small part, the amount of subnode space outside of the shaft is just small in comparison to the length of the shaft.

An important observation is that the traversal performance and the memory consumption significantly depend on the values of the branching factor N and the maximal depth d. While the table shows only the best values for every data structure, it is observable that the results are different for the cases where the values for all data structures lead to the same resolution. One example for those values are a resolution of 512^3 for the uniform grid, a maximal depth of 9 for the octree and the values $N = 8$ and $d = 3$ for the N-tree and the line space. In those cases the size of the data strucutre and the performance of the traversal are way better for the N-tree in comparison to the uniform grid and the octree.

The main benefit of the N-tree comes with the usage of the line space. For this we evaluated the performance gain of the line space in comparison to the N-tree for different values of N and d. The results are shown in table 2. The evaluated test scene is the Bunny, but the results for the other scenes are similar and indicate the same results. In all test cases it is observable that the usage of the line space for a small value of N ($N < 5$) brings little to no benefit. The same applies to big values of N ($N > 10$). As explained above the reason for the former is that the shafts are wider if N is small and the amount of subnode space outside of the shaft is bigger in comparison to its length. While this is unproblematic for long shafts with a big value of N, the problem there is that the shaft loses the potential of prediction because of its length. One non-empty subnode is sufficient to mark the corresponding shaft, so that the traversal needs to check all subnodes . It is observable that big values of d may not make any difference in performance. The reason for this is that the geometry is sufficiently stored in higher nodes and therefore the maximum depth of d is not needed. Moreover, if the values of N and d are too big ($N > 10, d > 5$) the data structure is too memory consuming and therefore not usable. The benefit of the line space as well as the optimal choice of the parameters are scene dependant, but in all choices of parameters the usage of the line space results in better performance than the corresponding N-tree without line space.

5 CONCLUSION AND FUTURE WORK

We have presented a novel and effective extension to existing spatial data structures. First, the N-tree, a variation of the Octree, has been discussed. Based on this we introduced the line space as an advancement for the N-tree by taking directional visibility information into account. Algorithms for the set-up and the improved

Parameters		N-tree	LS	Δ
$N \to 5$,	time (s)	0,342	0,333	-2,7%
$d \to 3$	size (MB)	40,3	40,8	+1,1%
$N \to 5$,	time (s)	0,137	0,123	-9,9%
$d \to 4$	size (MB)	57,1	60,3	+5,7%
$N \to 5$,	time (s)	0,136	0,126	-6,9%
$d \to 5$	size (MB)	57,6	61,9	+7,5%
$N \to 6$,	time (s)	0,198	0,180	-8,8%
$d \to 3$	size (MB)	42,7	44,2	+3,6%
$N \to 6$,	time (s)	0,144	0,126	-12,9%
$d \to 4$	size (MB)	96,9	112,2	+15,7%
$N \to 6$,	time (s)	0,145	0,126	-12,9%
$d \to 5$	size (MB)	96,8	112,4	+16,0%
$N \to 7$,	time (s)	0,148	0,131	-11,6%
$d \to 3$	size (MB)	47,5	52,4	+10,4%
$N \to 7$,	time (s)	0,144	0,127	-11,9%
$d \to 4$	size (MB)	109,5	132,8	+21,3%
$N \to 8$,	time (s)	0,151	0,118	-21,9%
$d \to 3$	size (MB)	56,8	70,5	+24,0%
$N \to 9$,	time (s)	0,123	0,101	-18,2%
$d \to 3$	size (MB)	82,5	106,7	+29,4%
$N \to 10$,	time (s)	0,166	0,112	-32,9%
$d \to 3$	size (MB)	123,3	172,8	+40,1%

Table 2: Performance comparison between the N-tree without and with the usage of the line space (LS) for different parameter sets of N and d. It is shown that higher values for these parameters result in a bigger memory consumption but leads mostly to a smaller traversal time with the usage of the line space. The used scene is the Bunny as individual object, other scenes produce similar results.

traversal were shown. By using binary information for the possible emptiness of all shafts within one node we conclude whether it is necessary to test the subnodes of the current node or if we are able to skip them. This additional skip condition results in a notable speed-up for all shown test cases. From there on there exist multiple paths for further study.

The binary entries in the line space are enough for estimating whether a ray from one point to another might be intersected by scene geometry. With this information it is possible to compute approximated shadows without testing the scene geometry for intersection at all. This might be sufficient for shadow computations of non-primary rays. Even for primary rays the resulting error may become negligible with a high value of d. This technique might even be used in rasterization where the computation of soft shadows is a rather tough topic.

By using a counter instead of the binary entries within the line space the data structure can be updated during runtime and therefore it can possibly be fast enough to handle dynamic scenes in realtime. The counter is incremented for each object intersecting a shaft. Thus, the line space can efficiently be rebuilt by decrementing the counter if geometry is removed and by incrementing the counter if geometry is added.

An obvious option for faster set-up or better runtime performance is to port the data structure and the traversal to latest generation GPU architectures since many necessary tasks could benefit from parallel computation.

In this paper we presented that the line space as directional visibility data structure is able to improve existing spatial data structures. Moreover, in future work we try to extend the impact of directional visibility conditions to current state-of-the-art data structures like Bounding Volume Hierarchies. We think that some kind of line space structure could even improve these data structures resulting in a win in performance for latest generation ray tracing data structures.

Another attempt would be to not only save binary or integer information in the line space, but to save the list of candidates directly in the shafts instead of saving them in the nodes of the N-tree. This would result in several advantages. First, the candidates within the shaft can be sorted beforehand which would improve performance during runtime. Moreover, the traversal itself would work without the typical node structure based on voxels but rather based on shafts which is more accurate and efficient.

6 REFERENCES

[AK87] ARVO J., KIRK D.: Fast ray tracing by ray classification. In *ACM Siggraph Computer Graphics* (1987), vol. 21, ACM, pp. 55–64.

[ÁSK14] ÁFRA A. T., SZIRMAY-KALOS L.: Stackless multi-bvh traversal for cpu, mic and gpu ray tracing. In *Computer Graphics Forum* (2014), vol. 33, Wiley Online Library, pp. 129–140.

[AW87] AMANATIDES J., WOO A.: A fast voxel traversal algorithm for ray tracing. *Eurographics '87* (1987), 3–10.

[Bre65] BRESENHAM J. E.: Algorithm for computer control of a digital plotter. *IBM Syst. J. 4*, 1 (Mar. 1965), 25–30.

[BWW01] BITTNER J., WONKA P., WIMMER M.: Visibility preprocessing for urban scenes using line space subdivision. In *Computer Graphics and Applications, 2001. Proceedings. Ninth Pacific Conference on* (2001), IEEE, pp. 276–284.

[CW12] COHEN M. F., WALLACE J. R.: *Radiosity and realistic image synthesis*. Elsevier, 2012.

[DS97] DRETTAKIS G., SILLION F.: Interactive update of global illumination using a line-

space hierarchy. In *Proceedings of ACM SIGGRAPH* (Aug. 1997), Annual Conference Series, pp. 57 – 64.

[ED06] EISEMANN E., DÉCORET X.: Fast scene voxelization and applications. In *ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games* (Redwood City, United States, 2006), ACM SIGGRAPH, pp. 71–78.

[ED08] EISEMANN E., DÉCORET X.: Single-pass gpu solid voxelization for real-time applications. In *Proceedings of Graphics Interface 2008* (Toronto, Ont., Canada, Canada, 2008), GI '08, Canadian Information Processing Society, pp. 73–80.

[EG08] ERNST M., GREINER G.: Multi bounding volume hierarchies. In *Interactive Ray Tracing, 2008. RT 2008. IEEE Symposium on* (2008), IEEE, pp. 35–40.

[EVG04] ERNST M., VOGELGSANG C., GREINER G.: Stack implementation on programmable graphics hardware. In *Vision Modeling and Visualization 2004* (2004), pp. 255–262.

[FS05] FOLEY T., SUGERMAN J.: Kd-tree acceleration structures for a gpu raytracer. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware* (2005), ACM, pp. 15–22.

[Hav00] HAVRAN V.: *Heuristic ray shooting algorithms*. PhD thesis, Faculty of Electrical Engineering, Czech Technical University, Prague, 2000.

[HKH11] HAPALA M., KARLIK O., HAVRAN V.: When it makes sense to use uniform grids for ray tracing. *Proceedings of WSCG'2011, Communication Papers* (Feb. 2011), 193–200.

[HW94] HAINES E. A., WALLACE J. R.: Shaft culling for efficient ray-cast radiosity. In *Photorealistic rendering in computer graphics*. Springer, 1994, pp. 122–138.

[JW88] JEVANS D., WYVILL B.: Adaptive voxel subdivision for ray tracing.

[KA13] KARRAS T., AILA T.: Fast parallel construction of high-quality bounding volume hierarchies. In *Proceedings of the 5th High-Performance Graphics Conference* (2013), ACM, pp. 89–99.

[KK86] KAY T. L., KAJIYA J. T.: Ray tracing complex scenes. *SIGGRAPH Comput. Graph. 20*, 4 (Aug. 1986), 269–278.

[KM07] KAMMAJE R. P., MORA B.: A study of restricted bsp trees for ray tracing. *Symposium on Interactive Ray Tracing 0* (2007), 55–62.

[LSCO03] LEYVAND T., SORKINE O., COHEN-OR D.: Ray space factorization for from-region visibility. In *ACM Transactions on Graphics (TOG)* (2003), vol. 22, pp. 595–604.

[NS04] NAM B., SUSSMAN A.: A comparative study of spatial indexing techniques for multidimensional scientific datasets. In *Scientific and Statistical Database Management, 2004. Proceedings. 16th International Conference on* (June 2004), pp. 171–180.

[PGSS07] POPOV S., GÜNTHER J., SEIDEL H.-P., SLUSALLEK P.: Stackless kd-tree traversal for high performance gpu ray tracing. In *Computer Graphics Forum* (2007), vol. 26, Wiley Online Library, pp. 415–424.

[RSH05] RESHETOV A., SOUPIKOV A., HURLEY J.: Multi-level ray tracing algorithm. In *ACM Transactions on Graphics (TOG)* (2005), vol. 24, ACM, pp. 1176–1185.

[RUnL00] REVELLES J., UREÑA C., LASTRA M.: An efficient parametric algorithm for octree traversal. *Journal of Winter School of Computer Graphics 8* (2000), 212–219.

[Sam89] SAMET H.: Implementing ray tracing with octrees and neighbor finding. *Computers And Graphics 13* (1989), 445–460.

[SS92] SUNG K., SHIRLEY P.: Ray tracing with the bsp tree. In *Graphics Gems III*, Kirk D., (Ed.). Academic Press, 1992, pp. 271–274.

[SS10] SCHWARZ M., SEIDEL H.-P.: Fast parallel surface and solid voxelization on gpus. *ACM Trans. Graph. 29*, 6 (Dec. 2010), 179:1–179:10.

[TI08] THIAGO IZE INGO WALD S. G. P.: Ray tracing with the bsp tree. *IEEE Symposium on Interactive Ray Tracing* (2008), 159–166.

[WK06] WÄCHTER C., KELLER A.: Instant ray tracing: The bounding interval hierarchy. *Rendering Techniques 2006* (2006), 139–149.

[WPS*03] WALD I., PURCELL T. J., SCHMITTLER J., BENTHIN C., SLUSALLEK P.: Real-time ray tracing and its use for interactive global illumination. *Eurographics State of the Art Reports 1*, 3 (2003).

ISSN 2464-4617 (print) WSCG 2016 - 24th Conference on Computer Graphics, Visualization and Computer Vision 2016
ISSN 2464-4625 (CD-ROM)

Real-time voxel rendering algorithm based on Screen Space Billboard Voxel Buffer with Sparse Lookup Textures

Szymon Jabłoński
Institute of Computer Science
Warsaw University of Technology
ul. Nowowiejska 15/19
00-665 Warsaw, Poland
s.jablonski@ii.pw.edu.pl

Tomasz Martyn
Institute of Computer Science
Warsaw University of Technology
ul. Nowowiejska 15/19
00-665 Warsaw, Poland
martyn@ii.pw.edu.pl

ABSTRACT

In this paper, we present a novel approach to efficient real-time rendering of numerous high-resolution voxelized objects. We present a voxel rendering algorithm based on triangle rasterization pipeline with screen space rendering computational complexity. In order to limit the number of vertex shader invocations, voxel filtering algorithm with fixed size voxel data buffer was developed. Voxelized objects are represented by sparse voxel octree (SVO) structure. Using sparse texture available in modern graphics APIs, we create a 3D lookup table for voxel ids. Voxel filtering algorithm is based on 3D sparse texture ray marching approach. Screen Space Billboard Voxel Buffer is filled by voxels from visible voxels point cloud. Thanks to using 3D sparse textures, we are able to store high-resolution objects in VRAM memory. Moreover, sparse texture mipmaps can be used to control object level of detail (LOD). The geometry of a voxelized object is represented by a collection of points extracted from object SVO. Each point is defined by position, normal vector and texture coordinates. We also show how to take advantage of programmable geometry shaders in order to store voxel objects with extremely low memory requirements and to perform real-time visualization. Moreover, geometry shaders are used to generate billboard quads from the point cloud and to perform fast face culling. As a result, we obtained comparable or even better performance results in comparison to SVO ray tracing approach. The number of rendered voxels is limited to defined Screen Space Billboard Voxel Buffer resolution. Last but not least, thanks to graphics card adapter support, developed algorithm can be easily integrated with any graphics engine using triangle rasterization pipeline.

Keywords

Computer graphics, voxel rendering, sparse voxel octree, sparse texture, point cloud, geometry shader, billboarding

1 INTRODUCTION

Voxel representations and rendering algorithms are one of the most extensively studied subjects in the field of computer graphics. For many years, voxels have been used in the visualization and analysis of medical and scientific data such as MRI scans [Potts04]. Nowadays, voxels representations are widely used in many fields of computer science, engineering and computer graphics, with applications ranging from fluid simulation to digital sculpting tools. However, because of high memory consumption and rendering complexity, their usage was limited to non-real-time graphics engines.

Thanks to increased computation power of today's GPUs and newly developed techniques, it seems that voxel-based representations are ready for real-time applications. Cyril Crassin was able to perform visualization of global illumination based on sparse voxel octree (SVO) and voxel cone tracing [Crassin11]. There are also a few promising implementations of efficient ray tracing of SVO [Laine10] and even object animation and deformation in real-time [Bau11, Wil13].

Computing performance is one of the most important measurements of real-time computer graphics algorithms. SVO ray tracing implementations showed that in this case ray tracing approach is much faster than triangle rasterization. Ray tracing rendering is scalable with screen resolution with the fixed cost of rendering, independent of the virtual scene complexity. Unfortunately, ray tracing does not have direct support from graphic accelerators and popular graphics APIs.

In the case of triangle rasterization pipeline, it is easy to overfill vertex shader invocations by redundant geometry data. We developed an algorithm which solves

that problem for voxel visualization with triangle rasterization pipeline. By selecting render candidates and filling a fixed size buffer prior to rendering process, we are able to limit voxel shader invocations to a defined maximum number. Thanks to that we achieved comparable or even better rendering performance results in comparison with SVO ray tracing approach. Moreover, we gain the support of modern hardware graphics APIs and rendering algorithms. Last but not least, the developed algorithm can be easily integrated with any popular game engines and used in video games.

2 RELATED WORK

There is a wide selection of literature on visualizing voxel objects. Over the years, many methods of direct and indirect voxel rendering have been developed. However, only a few of them are actually using a polygonal representation of the voxel structure in rendering process. We will focus on papers that are most directly related to our work.

One of the oldest and most citied method is Marching Cubes, presented by Lorensen and Cline in 1987 [Lorensen87]. The idea is to extract a polygonal mesh of an isosurface from a 3D discrete scalar field. Marching Cubes implementations are mainly used in the field of medical visualizations and special effects with what is usually called metasurfaces. There are also a few improvements of the base algorithm, like dual contouring [Ju02]. However, the marching cubes approach does not generate satisfying results in visualization of voxelized 3D objects with a lot of textures, like normal maps, ambient occlusion maps etc.

Splatting is one of the most studied methods of direct volume rendering. It was originally introduced by Westover [Westover89] The basic algorithm projects each voxel to the screen and integrate it into an accumulating render target. Using a painter's algorithm, it solved the hidden surface problem by visiting the voxels in either back-to-front or front-to-back order. Splatting is a perfect example of an object-order algorithm in contrast to ray-casting, which is an image-order algorithm. For years, the splatting technique has been used to render volumes of various grid structures [Westover89, Mao96, Westover91]. Martyn introduced a novel approach to realistic real-time rendering scenes consisting of many affine IFS fractals [Martyn10]. The implementation based on splatting and hardware geometry instancing makes it possible to achieve efficient visualization with small memory requirements.

Another interesting work in the field of voxel visualization is the particle-based approach presented by Juckel and Beckhaus [Beckh07]. The authors developed visualization of 3D scalar field by using a particle system. They proposed a unique method for rendering complex shapes as fuzzy or diffuse objects inside virtual environments. The algorithm converts surface geometry into the voxel-like grid that specifies the appearance of the shape. Using GPU implementation, they achieved rendering of dynamic objects inside a voxelized surface geometry. Particle systems were designed to handle millions of simple objects perfectly characterize voxel structures.

Although all of the presented methods propose interesting ideas related to visualization of the voxels, only ray tracing approach is able to render realistic 3D objects. As we mentioned before, the SVO ray tracing approach is the current standard of voxel visualization. Advantages of this algorithm are scalability with the screen resolution and fixed rendering cost resulting from a constant ray count. We have developed a voxel rendering algorithm which offers similar advantages and uses the triangle rasterization pipeline.

Using a newly available 3D sparse texture and highly optimized ray marching approach, we perform filtering of visible voxels. Then we fill screen size voxel buffer with filtered voxels data. Finally, we render voxelized objects with triangle based pipeline. We have managed to achieve efficient rendering performance results with a fixed rendering cost.

3 VISUALIZATION ALGORITHM

In this section, we describe our approach to visualize voxelized objects with the use of geometry shaders and deferred rendering. The features of our algorithm are as follows:

- Efficient rendering of high-resolution voxelized objects.
- Support for both static and dynamic objects.
- Representation based on SVO.
- Minimization of memory consumption.
- GPU acceleration for geometry generation and rendering.

3.1 Voxel representation

Voxels are simply a 3D generalization of pixels. Each value on a regular grid stores information such as color, normal vector or density. One of the most significant disadvantages of voxels compared to polygons is the memory consumption for high-resolution grids. One of the possible solutions to deal with this issue is the use of SVO. It eliminates empty spaces. Moreover, it gives the hierarchical level of detail (LOD) information about the source object. Fig. 1 presents example of SVO structure visualization.

The simplest way to visualize a voxel is to render a cube in 3D space. In such a representation, one would need to store data for twelve indexed triangles with

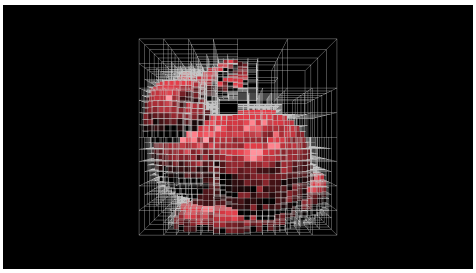


Figure 1: SVO structure created for sample object.

appropriate attributes. We could create proper data buffers on GPU and fill them with voxelized object data. Unfortunately, this is not an efficient solution for dynamic buffers. Also, we are still facing the problem with high memory consumption. Analyzing the rendering result of voxel representation with 3D cubes, we can realize that voxel looks same when viewed from different angles. Thus, we can visualize voxels as quads always faced to the viewer using billboarding approach [Behren05, Decaudin09].

Implementation based on quads instead of cubes significantly reduces memory requirements. However, it means that we need to use some kind of hardware geometry instancing or geometry generation method in order to render our object.

3.2 Geometry generation with shaders

Modern graphics APIs like OpenGL or DirectX offer a few ways to implement geometry generation with GPU programmable pipeline. The most straightforward way to implement our algorithm is to use geometry instancing functions. With this choice, we need to store our source quad in GPU memory and create data buffers for billboard attributes like positions and normal vectors. Geometry instancing is a very efficient method with data streaming functionality for handling dynamic objects. However, there is no way to control which quads will be generated and rendered on GPU. We can solve this by using streaming data buffers and performing calculations on CPU, but this is computationally expensive solution.

An alternative solution which we have implemented is based on the usage of programmable geometry shaders. In this method, we do not need quad representation data. We store our voxelized object as a collection of points and generate billboard quads on GPU using geometry shaders. Fig. 2 presents an example of rendering a 3D object based on a surface approximation with billboards.

As a result of using geometry shaders, we gain control on quads generation stage which, for example, gives us the possibility to execute the back-face culling algorithm or control billboards shape generation independently.

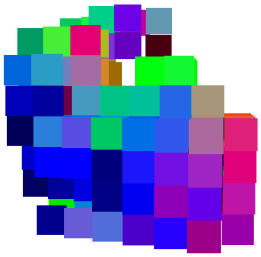


Figure 2: Visualization of low LOD object with collection of billboards.

3.3 Smooth shading realization

In the case of 3D objects based on polygonal representation, smooth shaded visualization can be achieved, for example, by using the Blinn-Phong normal interpolation model [Foley90]. Unfortunately, a voxelized surface has no information about adjacent voxels in the rasterization pass. This problem is typically solved by using the volume ray tracing algorithm which can be implemented on GPU with SVO structures. In order to achieve smooth shaded objects, we propose a new method based on the screen space approximation of voxel attributes data. Our smooth shading algorithm is based on multipass deferred rendering. In G-buffer generation pass we render voxel attributes data to floating point render targets.

The direct usage of rendered voxel attributes in the deferred composition pass with additional information about scene lighting produces blocky, flat shaded visualizations. In order to achieve smooth shaded visualization, we perform data interpolation by a filtering data texture with the 3x3 kernel Gaussian blur shader. Fig. 3 presents an example of voxel normal attributes interpolation in screen space.

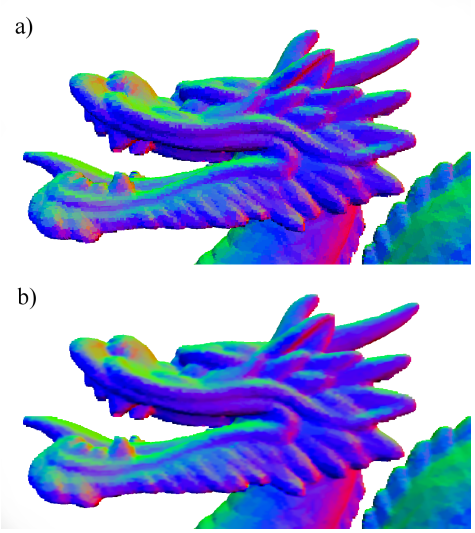


Figure 3: a) Object rendered without normal attribute interpolation b) Object rendered with normal attribute interpolation.

3.3.1 Smooth filtering control

In order to achieve a proper smoothing for all voxelized objects on the virtual scene, we must control the Gaussian blur intensity. The objects that are closer to the observer should be filtered stronger than the objects that are far away from the observer. Actually, we need to implement filtering control based on a similar manner as the LOD management algorithms [Lueb02].

In the proposed algorithm, we use the Gaussian blur with a 3x3 kernel as a voxel attribute filtering method. The most straightforward way to achieve stronger filtering is by changing the size of the blur kernel. However, it is a major waste of GPU computing resources, especially when we implement dynamic loops in our filtering shaders. We propose to use a filter shader sampler offset in order to achieve efficient and visually acceptable results.

The most commonly used LOD evaluation parameter is the distance [Lueb02]. We decided to use this parameter to control the smoothing intensity. Equation 1 presents a filter intensity calculation formula.

$$I = max\Big(0, min(\frac{MaxI}{(Dist - MinDist)^2}, MaxI)\Big) \qquad (1)$$

where:

I	= intensity of filter (sampler offset)
$MaxI$	= maximum intensity for the closest objects
$Dist$	= distance between object and observer
$MinDist$	= defined minimum distance

The result of the equation must be clamped to *<0, MaxI>*. Zero as a minimum value means that the object that is far away from the observer does not need to be filtered. The maximum intensity value must be defined by the user depending on virtual scene construction, as well as the minimum distance between an object and the observer, where voxel attributes are filtered with the maximum intensity.

3.3.2 Pixel depth based smoothing

Voxel attributes smoothing is done with screen space shaders. We cannot calculate the distance to each object on the virtual scene and pass it to the shader as a uniform value. Also, we cannot perform multiple filtering passes in order to fit in the time requirements of real-time graphic engines. In order to perform a distance-based data filtering, we use depth a buffer from G-buffer pass.

Depth buffer stores the depth of a generated pixels. In order to perform a smoothing operation with equation 1, we need to calculate a pixel position in world space. Using the inverse of the view projection matrix, we can reconstruct the pixel position in world space [Wright10]. By using the obtained value with a camera position transferred to shader code as a uniform, we are able to use the proposed equation and achieve a proper, distance based voxel attribute smoothing.

4 SCREEN SPACE BILLBOARD VOXEL BUFFER

In this section, we describe our approach to select voxel render candidates from an object voxel point cloud data in order to render the high-resolution object with a fixed size data buffer.

In order to limit the number of voxels required for rendering, we need to filter the voxel point cloud that would fill the screen space data buffer. Filtering operation can be done in a wide selection of methods. Voxel point cloud can be projected on the screen and by using depth test, render candidates can be selected. Another possible solution is to perform hierarchical occlusion queries in order to find visible SVO nodes. However, our main goal was to develop an algorithm with screen space computation complexity which will offer comparable performance results as the ray tracing. In order to select render candidates we developed ray marching algorithm with 3D lookup texture.

4.1 Filtering algorithm components

The smallest part of the screen is one pixel. It means that the image covers the maximum information when all pixels are filled by exactly one independent voxel. However, a 3D object can be represented by much more voxels than is needed to fill a render target. It is the biggest disadvantage of the graphics representation with polygons. Vertex shaders can be invoked for the data that would not fill any result image pixels or pixel overdrawn can cause an enormous performance hit. This problem does not occur in the context of rendering with ray tracing approach. For this reason, we decided to develop an efficient way to select object voxels that would fill the resulting image.

Our algorithm uses the structure which we called Screen Space Billboard Voxel Buffer. The algorithm is based on the three components:

- **Screen Space Billboard Voxel Buffer** — a fixed size data buffer which contains voxel data used in the rendering process. The size of the buffer corresponds to the render target resolution. Voxel filtering algorithm selects render candidates and fills voxel buffer data.
- **Voxel point cloud** — voxels data from the object's selected LOD. Each point stores information such as position, normal vector, texture coordinates and optional object id.

- **Sparse Lookup Texture** — 3D texture for voxel point cloud lookup table. With ray marching approach, we get an id of render candidate voxel. Using that id, voxel data is copied from voxel point cloud buffer to screen space voxel buffer. Thanks to the modern graphics APIs, we can create the texture that is much bigger than available memory and fill only selected *pages* of the texture [Wright10]. Object LOD control is managed by using sparse texture mipmaps to store 3D lookup table for different object LODs.

4.2 Voxel Filtering algorithm

In this section, we describe base steps of voxel filtering algorithm. The developed algorithm is based on the 3D sparse lookup texture ray marching.

Using ray intersection test with 3D texture, we are able to efficiently filter voxel cloud. However, this approach creates two potential problems. Firstly ray marching requires a lot of texture sampling operations. In order to achieve efficient performance results, it is required to implement a few optimization techniques like Object Order Empty Space Skipping [RezkSal09, Vidal08]. Secondly, standard 3D texture will require a lot of the VRAM memory. We solve this problem with the newly developed sparse texture from modern graphics APIs [Wright10]. From OpenGL 4.4 specification, AMD sparse texture extensions developed by Graham Sellers is available by GL_ARB_sparse texture. Additionally, OpenGL 4.5 specification added a new version of this extension with full shader side control.

4.2.1 Algorithm preparation steps

Voxel filtering algorithm can be divided into the data preparation and execution steps. The preparation steps are as follows:

1. Extract voxel points clouds from the required LODs. This step can be done in precomputation pass on virtual scene initialization.
2. Create 3D lookup table for voxel point cloud and store it in sparse texture. If we need to use LOD management of virtual scene, we store additional lookup tables in sparse texture mipmaps.
3. Create simplified 3D object triangle mesh based. This mesh will be used to optimize ray marching operation. It is important to create a polygonal mesh that vertices positions are in range of <-1.0,1.0>.

4.2.2 Algorithm execution steps

In the application rendering loop we perform algorithm execution steps as follows:

1. Perform visibility test of the 3D object with frustum culling and optionally occlusion culling tests. If our object is not visible or it is occluded by another object, the algorithm ends here for the selected object.
2. Render all visible objects simplified mesh offscreen. For all objects, we save normalized object space position in first render target and the object world space position in the second render target. The first texture will be used to perform Object Order Empty Space Skipping. The second will be used to handle scenes with numerous objects. Additional object id or LOD information will be saved in the same textures if needed.
3. Perform ray marching for all pixels using obtained textures as input. Using 3D sparse lookup texture, find first intersection and store voxel data in Screen Space Billboard Voxel Buffer. It is important to clear the current pixel screen space buffer data from the last frame in order to optimize rendering pass.
4. Render Screen Space Billboard Voxel Buffer using the algorithm described in section 3.

4.2.3 Algorithm conclusion and limitation

Using the fixed size voxel buffer we limited vertex shader invocations to the fixed number. Moreover, the voxel filtering algorithm based on 3D sparse texture ray marching can be implemented in a very efficient way. Thanks to that, we can render a virtual scene with numerous high-resolution 3D objects with triangle rasterization pipeline in real-time. It is impossible without filtering step with today's hardware.

The fixed size voxel buffer is a particularly efficient method in the case of high-resolution 3D objects. If an object is represented by more voxels that can be stored in Screen Space Billboard Voxel Buffer, the draw call operation can be significantly optimized by limiting vertex shader invocations. For example, Stanford Bunny object on 9 level of SVO is represented by about 2.5 million voxels. After the filtering pass, we need only about 140 thousand voxels to render the object. It means that even with the additional filtering pass and fixed cost rendering operation, performance results is noticeably better than rendering object without filtering pass.

5 IMPLEMENTATION DETAILS

In this section, we describe important implementation details of our algorithm. We have implemented our method using OpenGL 4.5 API with C++14 but there are no limitations to using any other graphics interface or programming language. All included shader source code listings are prepared in GLSL language. Due to the simplicity of the billboard based voxel representation, the presented algorithm can be easily implemented and integrated into all popular game engines. The only requirement is the support for programmable geometry and compute shaders.

5.1 Screen Space Billboard Voxel Buffer preparation implemenation

In this section, we describe *Screen Space Billboard Voxel Buffer* implementation. We will focus mainly on ray marching extensions and algorithm optimization steps.

5.1.1 Screen size static vertex buffer

The base component of the developed Screen Space Billboard Voxel Buffer algorithm is a static, fixed size data buffer for voxels data. Due to that we can create a static Vertex Buffer Object and fill it using compute shaders. Using Shader Storage Buffer Objects introduced in OpenGL 4.X API, we can bind the Vertex Buffer Object and access it from the compute shader. Therfor, both voxel point cloud input data and output Screen Space Billboard Voxel Buffer can be accessed on GPU. The layout of vertices data is the same as vertex layout that we use in the voxel visualization algorithm. The only potential difference is additional object id information if our virtual scene contains many independent 3D objects. In that case, an additional material data array is necessary to handle shading in rendering pass.

5.1.2 3D sparse texture

In our implementation, we used GL_ARB_sparse_texture and GL_EXT_sparse_texture2 extension according to test hardware specification. For a lookup table, *R32UI* internal format texture was used to store voxels id in the red channel. According to the texture internal format, the sparse texture is dived to the specified number of pages. We used 16x16x16 size pages for lookup table texture. If some page is empty, GPU will not allocate video memory for that page. On the shader side, we can check if sampled data is committed or not. If we cannot use the latest version of sparse texture extensions, according to the driver specification, sampling operation should return zeros.

5.1.3 Voxel filtering implementation

Voxel filtering algorithm was implemented with compute shader. Using ray marching approach, we seek for render candidates and then copy voxel data from point cloud to screen space voxel buffer. In order to optimize ray marching step, Object order empty space skipping pass was implemented. Using a prepared simplified triangle mesh, we render world and object space positions to off-screen render targets. Using saved pixel object space position, we can optimize ray marching by starting marching very close to the object surface. Additionally, we optimize ray marching for big, empty spaces. Figure 4 presents render results of example scene.

Created 3D sparse texture and pre-pass rendering results are used in final voxel filtering step. Listing 1 presents main parts of voxel filtering compute shader code. Ray marching and data preparation code have been omitted. Additionally, in order to simplify listing, some code was reduced to pseudocode or comments.

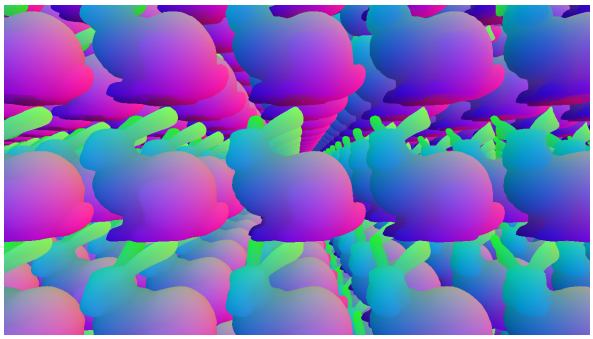


Figure 4: Ray marching start position from object order empty space skipping optimization pass.

```
layout (rgba16f) uniform image2D
    worldSpaceTex;
layout (rgba16f) uniform image2D
    objectSpaceTex;
uniform usampler3D lookupTex;

struct VertexLayout {
  vec3 position;
  vec3 normal;
  vec2 uv;
};

layout(std140, binding = 0) buffer VBO_Input
    {
  VertexLayout vbo_in[];
};

layout(std140, binding = 1) buffer
    VBO_Output {
  VertexLayout vbo_out[];
};

void main(void) {
  int id = pix.x*size.y + pix.y;
  vbo_out[id].uv.x = -1.0; // clear SSBVB

  vec4 posStart = imageLoad(objectSpaceTex,
      pix);
  if(IsEmpty(depth.a)) return;

  // perform ray marching fo defined sample
      count
  uint sample = texture(lookupTex,pos).r;
  if(IsResident(sample)) {
    int voxel = int(sample);
    vec4 off = imageLoad(worldSpaceTex, pix);
    // init SSBVB
    vbo_out[id].position =
        vbo_in[voxel].position;
    vbo_out[id].position += off.xyz;
    vbo_out[id].normal =
        vbo_in[voxel].normal;
    vbo_out[id].uv = vbo_in[voxel].uv;
  }
```

Listing 1: Voxel filtering compute shader.

5.2 Voxel rendering implementation

The major disadvantage of voxel representations is memory consumption. Using 3D textures to store object data is a major waste of the VRAM. For example, efficient usage of allocated memory in the case of Stanford Bunny surface voxelized 256^3 resolution is about 1.5%. The usage of SVO solves this problem. The SVO is a great method of voxel data compression based on optimizing empty and constant spaces. Additionally, we automatically gain an object LODs collection.

Ray tracing is the popular method of SVO rendering. It is efficient and produce great rendering results. However, in that method, full SVO data must be stored in the VRAM or data must be streamed from RAM to GPU memory. In order to easily handle the virtual scenes with many different objects based on the SVO, we used a different approach. Listings 2 - 4 presents a voxel rendering pipeline with geometry shaders.

```
layout (location = 0) in vec3 pos;
layout (location = 1) in vec3 normal;
layout (location = 2) in vec2 texCoords;

uniform mat4 modelView;
uniform mat3 invTModelView;

out VertexData {
  vec3 normal;
  vec3 position;
  vec2 texCoord;
} VertexOut;

void main() {
  gl_Position = vec4(pos,1.0);
  vec4 viewPos = modelView*vec4(pos,1.0);
  VertexOut.normal = invTModelView*normal;
  VertexOut.position = viewPos.xyz;
  VertexOut.texCoord = texCoords.xy;
}
```

Listing 2: Voxel rendering vertex shader

```
layout (points) in;
layout (triangle_strip, max_vertices = 4)
    out;

uniform mat4 projection;
uniform mat4 modelView;
uniform vec3 size;
uniform vec3 cameraPosition;

in VertexData {
  vec3 normal;
  vec3 position;
  vec2 texCoord;
} VertexIn[1];

out VertexData {
  vec3 normal;
  vec3 position;
  vec2 texCoord;
} VertexOut;

void main() {
  if(Backface() || EmptyScreenBuffer())
    return;

  vec4 center = modelView *
      gl_in[0].gl_Position;

  gl_Position = projection * (center + size);
  VertexOut.position = VertexIn[0].position;
  VertexOut.normal = VertexIn[0].normal;
  VertexOut.texCoord = VertexIn[0].texCoord;
  EmitVertex();
  // ... same operation for the rest
  EndPrimitive();
}
```

Listing 3: Voxel rendering geometry shader

```
uniform sampler2D tex;

in VertexData {
  vec3 normal;
  vec3 position;
  vec2 texCoord;
} VertexIn;

out vec4 albedoOutput;
out vec4 posOutput;
out vec4 normOutput;

void main() {
  vec3 norm = normalize(VertexIn.normal.xyz);
  albedoOutput = texture(tex,
      VertexIn.texCoord);
  posOutput =
      vec4(VertexIn.position,LinearizeDepth());
  normOutput = vec4(norm, 1.0);
}
```

Listing 4: Voxel rendering pixel shader

5.3 Screen space attributes smoothing integration

With developed algorithm, we are able to render a voxelized object alongside triangle objects and use any triangle rasterization pipeline algorithm. Unfortunately, a challenge appears when we need to implement the screen space data smoothing for selected voxel attributes. Using the deferred rendering pipeline we store all frame normal data in the G-buffer. Smoothing cannot affect triangle based objects because the information will be lost.

We developed an integration method based on stencil buffer. Using the stencil test we can exclude triangle-based objects from voxel objects. The algorithm based on stencil buffer is as follows:

1. Create and attach stencil buffer to the G-buffer.
2. Setup stencil to write defined value to stencil when rendering triangle objects and a different value for the voxel objects.
3. Attach stencil buffer to filtering pass frame buffer.
4. Setup stencil test to pass only fragments related to the triangle objects.

5. Setup stencil test to pass only voxel objects and apply filtering shader to passed fragments.
6. Use smoothed attributes in deferred composition pass.

6 RENDERING AND PERFORMANCE TEST RESULTS

All depicted timings were obtained on Intel Core i5 2500K CPU with NVidia GeForce GTX 660 GPU. All algorithms were implemented using OpenGL 4.5 API with C++14 for Windows 10 64-bit. We used Stanford Repository models as a test object [Stanford11]. Table 1 presents performance, memory requirements and test results for static and dynamic streamed voxel objects. Figures 5 - 7 presents rendering results of proposed voxel rendering algorithm.

Without a doubt, today's standard and most studied method for visualizing SVO is based on using the ray tracing algorithm on GPU. For that reason, we performed performance tests and compared them with our algorithm. As a reference, we used the „Efficient sparse voxel octrees" implementation that is available online [Laine10]. Performance test results are presented in Table 2. The results show that our approach offers comparable or even better performance than ray tracing SVO visualization.

It order to test how our algorithm performs on modern hardware designed for computer games, we performed additional tests on PC with Intel Core i7 4790K CPU with NVidia GeForce GTX 980 GPU. We prepared objects voxelized in 2048^3 resolution. Figure 8 presents rendering results of high-resolution objects. Figure 9 presents performance test of the scene rendered with Screen Scene Billboard Voxel Buffer.

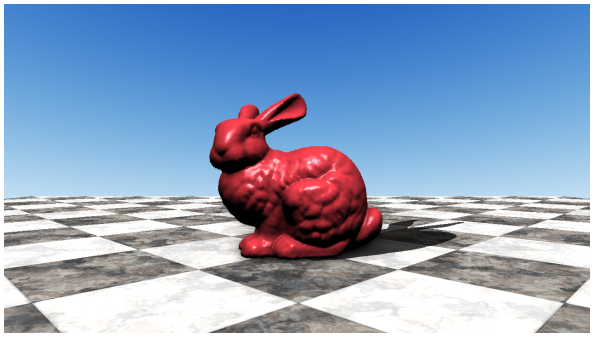


Figure 5: Stanford Bunny, 9 octree level.

7 CONCLUSIONS AND FUTURE WORK

In this paper, we presented a novel approach to efficient real-time rendering of numerous high-resolution voxelized objects with the fixed size *Screen Space Billboard Voxel Buffer*. The developed method can be used

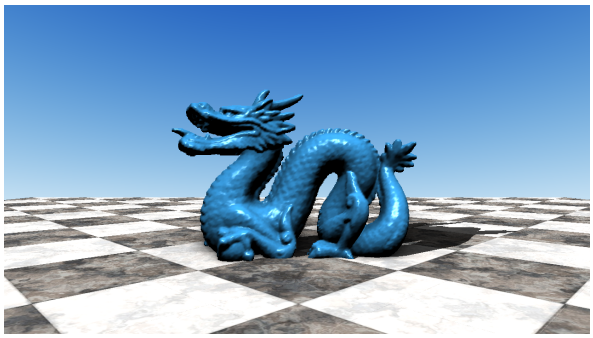


Figure 6: Stanford Dragon, 9 octree level.

Figure 7: Stanford Lucy, 10 octree level.

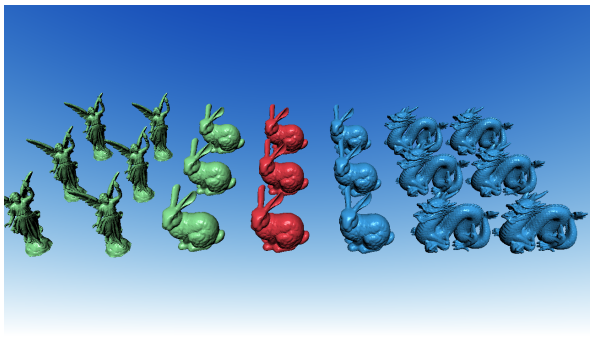


Figure 8: 21 test objects, 402 million voxels, 22 FPS achieved on 720p render target with GeForce GTX 980.

to render 3D objects represented by SVO with a standard triangle-based pipeline graphics engine. Thanks to the limitation of vertex shader invocations and the geometry shaders usage it is possible to achieve real-time rendering of billions of voxels. We achieved comparable or even better rendering performance results in comparison with the SVO ray tracing approach Moreover, our method is applicable to render both static and dynamic objects in real-time with the full support of modern hardware graphics APIs and rendering algorithms.

Used *Object Order Empty Space Skipping* efficiently optimized 3D sparse texture sampling. However, the current implementation is highly optimized for rendering non-occluding objects. We need to extend our ray marching implementation with additional ray traversal for potentially occluded pixels.

An obvious step forward would be an implementation of SVO traversal as a substitute for ray marching filter-

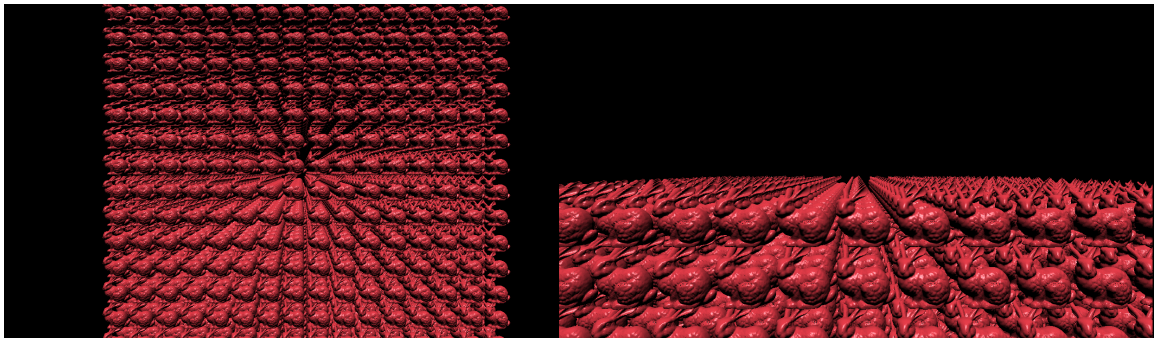


Figure 9: Rendering results using Screen Space Billboard Voxel Buffer in 720p with Nvidia Geforce GTX 980. Test scene contains 4096 Stanford Bunny which are represented by about 10 billion voxels (20 billion triangle in base algorithm). We achieved about 70 FPS for the left image and about 160 FPS for the right image.

Object	Octree level	Static object rendering time	Streamed object rendering time	Voxel grid file	Octree file	VRAM
Bunny	7	0.07 ms	0.11 ms	2.06 MiB	15.0 MiB	0.67 MiB
	8	0.26 ms	0.61 ms	8.30 MiB	60.9 MiB	0.67 MiB
	9	1.24 ms	3.75 ms	33.20 MiB	255 MiB	3.04 MiB
Dragon	7	0.10 ms	0.12 ms	2.81 MiB	22.2 MiB	0.92 MiB
	8	0.37 ms	0.95 ms	11.3 MiB	88.8 MiB	0.92 MiB
	9	1.81 ms	5.15 ms	45.3 MiB	363 MiB	4.26 MiB
Lucy	7	0.05 ms	0.11 ms	1.69 MiB	11.0 MiB	0.52 MiB
	8	0.20 ms	0.40 ms	6.86 MiB	45.3 MiB	0.52 MiB
	9	0.95 ms	2.73 ms	27.5 MiB	183 MiB	2.32 MiB

Table 1: Performance, memory requirements and test results for static and dynamic streamed voxel objects without using developed *Screen Space Billboard Buffer* algorithm. Render target resolution was 720p.

Object	Octree level	Resolution	Ray tracing render time	Ray tracing manage time	Ray tracing FPS	SSBVB render time	SSBVB manage time	SSBVB FPS
Bunny	7	720p	8.43 ms	10.20 ms	78	2.61 ms	1.09 ms	253
		1080p	15.16 ms	17.90 ms	48	6.80 ms	2.48 ms	102
	8	720p	8.80 ms	10.60 ms	76	2.56 ms	1.46 ms	235
		1080p	15.25ms	18.30 ms	47	5.92 ms	3.05 ms	106
	9	720p	9.32 ms	11.25 ms	72	2.59 ms	2.34 ms	192
		1080p	16.01 ms	19.21 ms	44	5.92 ms	4.63 ms	90
Dragon	7	720p	6.44 ms	8.20 ms	93	2.88 ms	1.96 ms	198
		1080p	11.10 ms	13.80 ms	60	7.64 ms	3.89 ms	83
	8	720p	6.89 ms	8.60 ms	89	2.82 ms	3.38 ms	153
		1080p	11.68 ms	14.20 ms	68	6.52 ms	6.74 ms	73
	9	720p	7.47 ms	9.22 ms	84	2.85 ms	5.78 ms	112
		1080p	11.92 ms	15.30 ms	54	6.50 ms	11.91 ms	54
Lucy	7	720p	4.51 ms	6.20 ms	115	2.59 ms	1.17 ms	252
		1080p	8.01 ms	10.50 ms	72	6.65 ms	2.38 ms	107
	8	720p	4.75 ms	6.45 ms	112	2.51 ms	1.86 ms	219
		1080p	8.30 ms	10.98 ms	69	5.67 ms	3.70 ms	101
	9	720p	5.04 ms	6.71 ms	109	2.49 ms	3.11 ms	170
		1080p	8.70 ms	11.30 ms	69	5.63 ms	6.13 ms	81

Table 2: Performance test comparison between developed *Screen Space Billboard Buffer* algorithm and ray tracing implementation [Laine10]. SVO ray tracing times are obtained from the performance tools included in the implementation.

ing approach. We can actually use sparse textures to store SVO. In that case, we will need to store full SVO in GPU memory. It will increase memory requirements and slightly impends dynamic object handling. However, voxel filtering could be faster and more precise in comparison with ray marching approach.

8 REFERENCES

[Bau11] Bautembach D., Animated sparse voxel octrees, Bachelor Thesis, University of Hamburg, 2011.

[Beckh07] Beckhaus S., and Juckel, T., Rendering diffuse objects using particle systems inside voxelized surface geometry, The 15-th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision2007 (WSCG 2007).

[Behren05] Behrendt, S., Colditz, C., Franzke, O., Kopf, J., and Deussen, O., Realistic real-time rendering of landscapes using billboard clouds, Computer Graphics Forum, 2005.

[Crassin11] Crassin, C., Neyret, F., Sainz, M., Green, S., and Eisemann, E., Interactive indirect illumination using voxel cone tracing, Computer Graphics Forum (Proceedings of Pacific Graphics 2011), vol. 30, no. 7, sep 2011.

[Decaudin09] Decaudin, P., Neyret, F., Volumetric Billboards, Computer Graphics Forum, Volume 28 (8), pp 2079-2089, 2009.

[Foley90] Foley, J. D., van Dam, A., Feiner, S. K., and Hughes, J. F., Computer Graphics: Principles and Practice (2Nd Ed.). Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1990.

[Ju02] Losasso, T. Ju, F., Schaefer, S., and Warren, J., Dual contouring of hermite data, ACM Trans. Graph., vol. 21, no. 3, pp. 339-346, Jul. 2002.

[Laine10] Laine, S., and Karras, T., Efficient sparse voxel octrees, in Proceedings of the 2010 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games, ser. I3D 2010. New York, NY, USA: ACM, 2010, pp. 55-63.

[Lueb02] Luebke D., Watson B., Cohen, J., D., Reddy, M., and Varshney, A., Level of Detail for 3D Graphics. New York, NY, USA: Elsevier Science Inc., 2002.

[Lorensen87] Lorensen, W. E., and Cline, H. E., Marching cubes: A high resolution 3d surface construction algorithm, SIGGRAPH Comput. Graph., vol. 21, no. 4, pp. 163-169, 1987.

[Mao96] Mao, X., Splatting of Non Rectilinear Volumes Through Stochastic Resampling, IEEE Transactions on Visualization and Computer Graphics, 2(2), 1996, pp. 156-170.

[Martyn10] Martyn T., Chaos and graphics: Realistic rendering 3d ifs fractals in real-time with graphics accelerators, Comput. Graph., vol. 34, no. 2, pp. 167-175, Apr. 2010.

[Potts04] Potts S., and Möller T., Transfer functions on a logarithmic scale for volume rendering, in Graphics Interface 2004, ser. GI 2004, 2004, pp. 57-63.

[Reinders07] Reiders, J., Intel Threading Building Blocks, O'Reilly Associates, Inc., 2007.

[RezkSal09] Rezk Salama, C., Hadwiger, M., Ropinski, T., Ljung, P., Advanced Illumination Techniques for GPU Volume Raycasting, ACM SIGGRAPH Courses Program, 2009.

[Stanford11] The Stanford 3D Scanning Repository, Stanford University, 22 Dec 2010, Retrived 17 July 2011.

[Vidal08] Vidal, V., Mei, X. Decaudin, P., Simple Empty-Space Removal For Interactive Volume Rendering, J. Graphics Tools, vol. 13, no. 2, pp. 21-36, 2008.

[Westover89] Westover, L. Interactive volume rendering. In: VVS '89: Proceedings of the 1989 Chapel Hill workshop on Volume visualization, New York, NY, USA, ACM (1989) 916

[Westover91] Westover, L.A., SPLATTING: A Parallel, Feed-Forward Volume Rendering Algorithm, Ph.D. Dissertation, Department of Computer Science, The University of North Carolina at Chapel Hill, 1991.

[Wil13] Willcocks, C. G., Sparse volumetric deformation, Ph.D. disseration, Durham University, 2013.

[Wright10] Wright, R., S., Haemel, N., Sellers, G., Lipchak, B., OpenGL SuperBible: Comprehensive Tutorial and Reference, Addison-Wesley Professional, 2010.

ISSN 2464-4617 (print) WSCG 2016 - 24th Conference on Computer Graphics, Visualization and Computer Vision 2016
ISSN 2464-4625 (CD-ROM)

Fracturing Sparse-Voxel-Octree objects using dynamical Voronoi patterns

Jakub Domaradzki
Institute of Computer Science
Warsaw University of Technology
ul. Nowowiejska 15/19
00-665 Warsaw, Poland
J.Domaradzki@stud.elka.pw.edu.pl

Tomasz Martyn
Institute of Computer Science
Warsaw University of Technology
ul. Nowowiejska 15/19
00-665 Warsaw, Poland
martyn@ii.pw.edu.pl

ABSTRACT

We introduce a new Voronoi-based method to fracture objects represented by sparse voxel octrees (SVOs). Our approach is inspired by the pattern-based methods, however, in contrast to them, it doesn't require pattern precomputation. Moreover, thanks to the octree structure, the surfaces of the fractured pieces of geometry are created efficiently and robustly. Every fracture pattern is unique and centered at the impact location. A novel islands detection technique is also provided, which is tunable to a desired level-of-detail accuracy. The fractured pieces, which are determined as a consequence of the object's destruction, are represented by individual SVOs, and treated and simulated as rigid bodies. For this purpose, we also propose a new collision detection technique, which extends the previous image-based methods to voxels. As a result, deep penetrations of colliding objects, resolved on various levels of physics that can be specified individually for each pair of the objects, are handled in parallel with no extra cost. In order to demonstrate our technique, a number of scenarios are presented, including a partial fracturing of objects with fine details.

Keywords

SVO, Voronoi decomposition, pattern fracturing, rigid body physics

1 INTRODUCTION

For many years voxels have been successfully used in lots of applications in computer graphics. From special effects up to medical imaging, we benefit from volumetric information delivered by voxels. Transparent, layered and with vague surfaces models are the main target for this object representation. There were many limitations that came along with voxels, such as large memory consumption and lack of hardware support, and some of them are still present nowadays. However, many new techniques have been developed, which made voxels more competitive than ever. With constant increase in computational power of modern graphics processing units, we may be facing situation, when methods based on voxels will gradually supersede the ones based on triangles.

In this context, probably one of the best promising concepts is Sparse Voxel Octree (SVO) [Cra11] — a hierarchical structure that lately has made voxels popular as a representation for solid objects in computer graphics. Less memory consumption, various levels of detail, and necessary access only to small subset of full voxel data are some benefits of SVO that have led to creation of realistic, high-resolution voxel models and large voxel environments. In this paper, we made an attempt to take advantage of this representation and shed a new light on the problem of objects fracturing.

Special effects of destruction are widely present in today's computer games and movie industry. Depending on a purpose, they can be either highly realistic but requiring lots of computational power, or fast but giving only an approximate illusion of observed phenomena. The latter approach is aimed for real-time performance, where accuracy is not as important, that this is usually the case in games. Many methods are used in order to do that in a fast but somewhat "fake" way, such as pre-fracturing of objects and replacing models when collision happens. Such an approach, however, requires preparations of lots of game assets. Recently, pattern-based fracture methods have been proposed for the mesh representation. They deal with this problem by decomposing the destruction process into two parts: the generation of a fracture pattern and then its multiple application on number of objects. While the first part can be done by an artist, the latter requires specialized techniques in order to robustly cut objects, mainly due to their geometrical complexity. Similarly, after the ob-

ject's destruction, the movement of fractured pieces is simulated and it also suffers from the same reason: the more detailed object, the harder to calculate collision.

We strongly believe, that representing objects with sparse voxel octrees is a remedy to all these problems. With spatial information about an object, built from basic geometric figures like boxes, many algorithms can be simplified. What is more, due to the SVO hierarchical structure, many calculations can be avoided and performed on different levels of details. Our main contribution include:

- a method to dynamic fracture with Voronoi decomposition, which can be applied locally;
- a method for detecting separated volumes in a SVO;
- an algorithm for collision detection, that adopts the image-based approach [FBAF08] to voxels.

2 RELATED WORK

Fracture Simulation. Fracture modeling for computer graphics greatly enhances physics simulation widely present in modern computer games and special effects in movies. Probably it wouldn't be so common today if it weren't for work by Terzopoulos et al. [TF88] and Norton et al. [NTB*91] which pioneered this area of study in computer graphics. O'Brien and Hodgins followed them and focused in their papers on simulating brittle [OH99] and ductile fracture [OBH02]. Their approach was based on the finite element method (FEM) used to compute internal stresses and fracture propagation directions. Unfortunately, cutting a mesh and its actualization, which is one of the most challenging issues in such approaches, are still present nowadays [WRK*10]. There are also methods that formulate the problem of fracture simulation as a quasi-static stress analysis [ZBG15]. They have the potential to be cheaper than a fully dynamic deformation simulation, but tends to produce deadened motions. An example of another interesting work can be [CYFW14], in which low resolution objects are enriched during fracture with extra details based on material strength field.

Although, from the "physical" viewpoint, the results of the mentioned methods can be perceived as acceptable, our main interest lies in solutions aimed for real-time systems. Since the pattern-base approach, at least conceptually, plays well with voxel representations and, moreover, is efficient and robust, we decided to utilize it in our SVO fracture method. A number of methods have been proposed to generate fracture patterns, such as the ones based on Voronoi diagrams [SSF09][BCC*11][MCK13] or engaging simulations [IO09]. In our approach, in contrast to the previous ones, we don't need any precalculated fracture pattern, as it is generated on-the-fly (like in [SO14]), when collision happens.

Collision Detection and Response. Collision detection itself covers broad area of investigation in computer graphics. Due to colliding objects geometrical complexity it is common to take advantage of hierarchical bounding structures, including bounding boxes [GLM96] and bounding spheres [Hub95]. Fortunately, in the case of SVO, objects are inherently represented hierarchically, so there is no need for any additional space searching optimizing structure.

More recently, some work on collision detection has been focused on exploiting GPUs, taking advantage of their inherent computational parallelism. Most of the presented techniques return pairs of colliding primitives delivering necessary information for objects separation. In particular, Layer Depth Images (LDIs) proved to be very useful for this goal [HTG04]. In this method, at the first stage, a broad phase collision check is performed, resulting in bounding boxes representing the intersection volume of boxes' pairs that enclose colliding objects. Then, the volumes are rendered to LDIs using GPU. Finally, iterating over LDIs along a chosen viewing axis, one can calculate the collision volume by inspecting the pairs of consecutive texels. In the original LDI method as presented by Heidelberger et al. only collision detection but not its response was taken into account. That was addressed later by Faure et al. [FBAF08] by delivering the derivatives at the vertices of the meshes of colliding objects in order to generate forces for minimizing the volume of collision. In addition to the penalty-based method used in [FBAF08], Allard et al. [AFC*10] proposed also the constraint-based one including Coulomb friction.

In the context of collision detection, our method can be viewed as an extension of that by Faure et al., and it can also be incorporated into the method by Allard et al. Our main contribution lies in the efficient determination of the collision area. The main advantage of our method is that we can generate collision information in parallel for all collided objects at accuracy specified individually for a given pair of objects. While Faure dealt with objects represented with meshes, we utilize SVOs. One should notice that this new problem formulation fits very well to the LDI-based solution by Faure.

Sparse Voxel Octrees. The pursuit of efficient exploitation of voxel structures as representations of solid objects in computer graphics lasts for years. Nowadays, with the aid of modern GPUs along with the new GPU-specialized programming techniques, the benefits of voxel model of solids become not only evident but even more and more spectacular. Probably the best example of this new life of voxels in computer graphics is Crassin's research on SVO [Cra11] — one of the notable results is a global illumination method based on SVO and cone tracing [CNS*11]. Moreover, there are also some work that focuses mainly on efficient raytrac-

ing over SVO [LK10], which shows that ray casting using SVO can be faster than when objects are represented with meshes. Although the majority of papers focus on visualization of static voxel models, there were also some approaches to animation of voxel models. Crassin proposed to animate objects by constant voxelization to SVO, which unfortunately requires object's input mesh and scales poorly. Nevertheless, the SVO animation can be realized in other ways. An example is the method developed by Bautembach [Bau11], in which SVO is animated only during visualization, without any influence on the structure of an input model. His approach was then extended by Willcocks [Wil13], who presented a method for volumetric deformation and animation of large number of objects in the scene. What is more, octrees can be built very fast using recent techniques presented in [ZGHG11][GPM11][Kar12]. If there is not enough memory in GPU to build SVO by means of the methods, one can use the out-of-core approaches described by [BLD14][PK15], which utilize only a fraction of memory required to store the model.

The above and inevitably incomplete set of citations show that the computer graphics community nowadays experiences a renaissance of voxels, which manifests in constant development of new voxel techniques and applications. Nevertheless, to the best of our knowledge, this paper is the first report on the application of SVO in the tasks such as physics of rigid body collisions and fracture simulation.

3 SVO STRUCTURE

Our SVO structure is very similar to the one used by Crassin et al. [Cra11]. First of all, there is an octree for nodes' descriptors, represented as two 32 bit integers. Each node descriptor has the information about whether the node: is terminal (1 bit), represents empty space (1 bit), is internal (1 bit). The remaining 29 bits of the first integer are used as a pointer to the descriptor of the first of node's eight children. What is more, with each descriptor there is associated another pointer, as a second integer, to the densely packed voxel's data: color and normal vector.

3.1 Voxel Types

In our SVO structure, there are three kinds of nodes: boundary, internal and empty. *Boundary nodes* are the nodes that represent the boundary of the object and they are used in visualization. If the SVO is derived from a mesh, then the nodes are created during voxelization process and contains information about color and normal vector. They can be expanded further up to the finest SVO level. One of their child nodes can be an empty node. The *empty node* indicates that its volume doesn't intersect the object's surface, that is the node lies either inside or outside of the object — it is always a terminal node. There are also *internal nodes* that, from the algorithmic point of view, can be considered as a special case of empty nodes, because an empty node is flagged as internal when it passes the test (Sec. 3.2) whether it is located inside of the object. As such, internal nodes represent the interior of the object. These nodes are very important as they deliver the information that is necessary for objects fracturing and physics calculations.

3.2 Internal Nodes Test

After the SVO creation, the test for internal nodes is conducted for each empty node on every SVO level, because empty non-internal nodes are skipped during the fracturing process and collision detection. The test is based on ray casting. A ray is shot from the empty node center and has the same direction as the normal vector of the parent node. Next, the ray traverse the SVO down to its leaves. The two outcomes of this operation are possible. Either the ray doesn't hit anything or it encounters one of the leaves. In the first case, the internal node test has failed. In the second case, an additional test must be performed to determine which side of the leaf node was hit. The test rely on the comparison of the ray direction and the node normal vector. If the angle between them is smaller than 90 degrees, then the internal node test passed and the node is assumed to belong to the interior of the object (in other case, the empty node lies outside the object, so the test failed).

4 FRACTURING SVO OBJECT

4.1 Fracture Algorithm

In this section we outline our algorithm for fracturing SVO objects.The underlying assumption is that a SVO object is cut into pieces with a fracture pattern represented by a space-partitioning structure that divides the 3D Euclidean space into convex regions (Sec. 4.2). As such, the fracture pattern consists of (planar) faces determining the slicing areas which are then used to partition the SVO object.

The main part of the algorithm is to determine subsets of the SVO voxels that represent the surfaces of fractured pieces at the accuracy of the SVO highest level. The voxels that constitute the subsets are the boundary voxels (Sec. 3.1) from the SVO leaves as well as internal voxels that are intersected by a pattern face at the SVO highest level — we call the latter voxels the *HL internal voxels*. One should note, however, that internal voxels do not have children in the SVO, so the HL internal voxels that are not children of boundary voxels are not physically present in the original SVO. Therefore the HL internal voxels have to be dynamically created during the fracturing process.

In order to determine the HL internal voxels the SVO is traversed from the root to the leaves, and at each SVO

level the intersections of voxels with the pattern faces are tested (Sec. 4.3). If an internal voxel is intersected by a pattern face and the voxel is not the HL internal voxel, then its eight child internal voxels are dynamically created at the next SVO level. With regard to the voxel-face intersection tests the two following facts should be pointed out. First, the necessary condition for a child voxel to be intersected with a pattern face is the presence of the intersection of the face with the voxel's parent. Secondly, the parent of an internal voxel may be an internal voxel (in this case the child voxel was dynamically created) or a boundary voxel. Therefore at each SVO level the voxel-face intersection tests are performed for both interior and boundary voxels whose parent voxels have been intersected at the previous SVO level (Fig. 1).

Having the subsets of voxels, we regard the voxels as the boundary voxels of the fractured pieces (Sec. 4.4) represented by the subsets, and for each subset we build a SVO (Sec. 4.6) on the basis of the subset's component voxels treated as the SVO leaves.

One should note, however, that the SVO partition procedure does not guarantee that the resulting subsets of voxels will be connected sets. It may happen that a subset can be partitioned into two groups of voxels such that for all voxels in one group there is no adjoint voxel in the other group. In such a case the subset represents two (or more) disjoint fracture pieces which should be treated as individual objects by a physics engine. Such autonomous groups of voxels within a single subset we call *islands*. In order to detect them, the additional test (Sec. 4.5) has to be performed before the SVO creation.

4.2 Voronoi Fracture Pattern

Patterns used to fracture objects can be obtained in many ways. Usually, it is a precomputed decomposition of space, based on Voronoi diagrams. During the fracture process performed at run-time, the pattern is aligned with the target object at an impact location and properly rotated. Such an approach was recently presented by Sue et al. in [SSF09]. In their method, the fracture pattern is applied to a mesh object using the level set-based approach, which requires high resolution grids to get thin features and, consequently, it is both computationally expensive and memory consumptive. On the other hand, the fracture pattern may be applied directly to a mesh. However, a naive application of this idea would be cumbersome to implement robustly. To deal with it, Müller et al. [MCK13] implement a fracture pattern as a set of meshes, and a mesh to fracture is initially split to convex pieces with the use of the Volumetric Approximate Convex Decomposition (VACD). As a result, they are capable to locally destruct mesh objects in real-time.

In contrast to the previous methods, in our approach we represent a fracture pattern by a finite set of 3D points

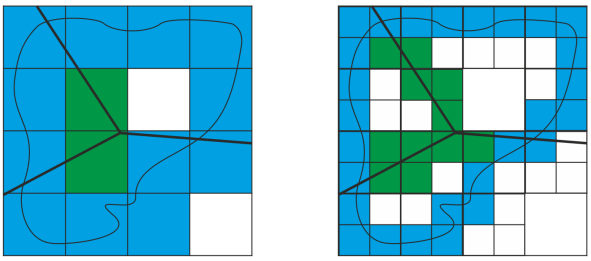


Figure 1: Voxels located on both the object's boundary and the Voronoi faces are expanded into child nodes on the next SVO level. The voxels marked with blue color lie on the boundary, while the green ones are internal, generated on-the-fly.

which are treated as seeds of a Voronoi diagram used to fracture a SVO object and generated dynamically at run-time. To deliver the impression that the fracture concentrates around the impact location, the Voronoi seeds are generated at random on a set of spheres of growing radii and centered at the point of impact. By manipulating the length of the radii we can obtain a variety of patterns and different sizes of fractured pieces. Of course, there is also a possibility to locate the seeds in a more physical way, which would make the outcome of the fracture process even more realistic.

4.3 Voxel-Pattern Faces Intersection Test

One of the main operations of our algorithm for fracturing SVO objects is the voxel-pattern faces intersection test which is performed at each level of the SVO as described in Sec. 4.1. The intersection test does not require the determination of the faces of the Voronoi diagram since it is realized directly on the basis of the set of the Voronoi seeds.

Let $S = \{1,\ldots,n\}$ be the set of the indices of the seeds $\{s_i\}_{i\in S}$ defining a Voronoi fracture pattern. Define a function $\gamma : \mathbb{R}^3 \to 2^S$ such that

$$\gamma(x) = \{k \in S : \|s_k - x\| = \min_{i\in S} \|s_i - x\|\}. \qquad (1)$$

Since there is the one-to-one correspondence between the Voronoi seeds and the Voronoi cells, given a point $x \in \mathbb{R}^3$, the function γ determines the set of the indices of the Voronoi cells that include x. (The resulting set of indices is not a singleton, if x lies on a face, an edge, or a vertex of the Voronoi diagram).

Now, we can move to the voxel domain and define a function that, given a voxel specified by the set $V = \{v_i\}_{i=1,\ldots,8}$ of its vertices, maps the voxel into the set of the indices of the Voronoi cells the vertices are located in:

$$\upsilon(V) = \bigcup_{v\in V} \gamma(v). \qquad (2)$$

Since the voxel V is intersected by a face of the Voronoi pattern if V has a (nonempty) intersection with at least two Voronoi cells, it is easy to see that the intersection

exists if the set of indices given by $\mho(V)$ is not a singleton. Of course, in a practical implementation of the intersection test we can stop computing the set $\mho(V)$ if its subset obtained in an intermediate step contains at least two indices.

4.4 Fracture Boundary Set

As a result of the voxel-pattern faces intersection tests performed while traversing the SVO, we obtain an unordered set of the HL internal voxels (see Sec. 4.1). The set together with the set of the SVO leaf boundary voxels constitute the set $\mathscr{B}$ of voxels which are the boundary voxels representing the surfaces of the fractured pieces at the accuracy of the SVO finest level. In order to assign these boundary voxels to the adequate pieces we should decompose the set $\mathscr{B}$ into the *disjoint* family of the connected subsets of voxels that represent, first of all, the pieces induced by the Voronoi cells, and then islands within a piece, if the piece proves to be a disconnected set.

Nevertheless, the voxels in $\mathscr{B}$ that have a nonempty intersection with a pattern face belongs to two or more Voronoi cells of the pattern and, as such, they should be "divided" between the cells. This leads us to the *fracture boundary set* (FBS) which is constructed from $\mathscr{B}$ by computing for each $V \in \mathscr{B}$ the set of indices $\mho(V)$, creating copies of V in a number equal to the cardinality of $\mho(V)$, and assigning to each copy the consecutive index from $\mho(V)$. Since the indices from $\mho(V)$ determine the Voronoi cells that V belongs to (Sec. 4.3), the copies of a voxel in FBS are treated as disjoint sets from the point of view of the voxel space partitioned with the fracture pattern. The FBS is the set on which we carry out the operations of grouping voxels into connected sets of fractured pieces — the algorithm described in the next section.

Apart from an index of the Voronoi cell, each voxel in FBS has to store attributes required for visualization, i.e., color, position, and normal vector. The attributes are determined on the basis of the attributes of original voxels from the SVO. The voxel's position is computed during the SVO traversal on the basis of the location of the voxel within the SVO. The remaining two attributes depend on whether the original voxel is a boundary voxel or a HL internal voxel. The copies of a boundary voxel inherit a color and normal vector from their original. In the case of the copies of a HL internal voxel the color may be obtained from a volumetric texture or it can be generated procedurally. In turn, their normal vectors are determined on the basis of the normals of the Voronoi faces that intersect the HL internal voxel. If the HL internal voxel does not lie on an edge or a vertex of the intersecting Voronoi face, the normal for the copy of the voxel in FBS is just equal to the face normal pointing outside the Voronoi cell associated to the copy; otherwise the normal is computed by averaging the normals of all faces that intersect the HL internal voxel.

4.5 FBS connected-component detection

Since FBS is just an unordered set of boundary voxels, we have to group the voxels into the connected subsets of the individual fractured pieces, i.e., the pieces induced by the Voronoi cells and eventually the islands within a piece, if the piece is a disconnected set (Sec. 4.1). However, before the actual voxel grouping takes place, in the initial step, FBS is sorted by the voxel positions represented by Morton codes [Kar12] (Fig. 2). Thanks to that, the voxels can be addressed by one-dimensional index, which is essential for the island detection algorithm as well as later, for constructing the SVOs (Sec. 4.6).

Now, we obtain the FBS voxels grouped with regard to the fracture pattern cells, just by carrying out the stable sorting by the $\mho(V)$ indices, which were assigned to the voxels during the FBS construction (as described in the previous section). As a result, the groups are placed one after another in linear memory, and the voxels within each group are ordered by Morton code. We call the groups of voxels *Voronoi cell pieces* (VCPs). In the next step each of the VCP groups has to be checked for connectivity, that is the island detection test is performed.

4.5.1 Island detection algorithm

We assume that a voxel is connected with another voxel from its nearest neighborhood if the voxels share a face, an edge or a vertex, so we assume 26-connectivity. Our solution to the island detection problem is based on the well-known conception of *connected-component labeling* (CCL — see e.g. [SB10]). The CCL underling idea is to label the elements of an input set with consecutive numbers, and then, to maximize iteratively the labels in the range of the elements' adjacent neighbors. This way, for each connected subset, the maximum label in the subset propagates between the subset's elements. At the end of the procedure the connected subsets are distinguished from one another by a maximum label that is associated to each element of a connected subset.

Although there are a number of efficient CCL algorithms, both sequential and parallel ones, to our best knowledge, all of them assume and operate on the input data organized in a regular structure such as a 2D uniform grid of pixels or a 3D uniform grid of voxels. Therefore the necessary neighborhood information for each basic structure element is easily accessible. Unfortunately, this is not the case for SVO and, as a consequence, for FBS, since the voxel neighborhood information does not follow from the ordering of elements inherent for these structures. Of course, one could augment the structures in the additional information by storing the pointers to neighbors of each voxel,

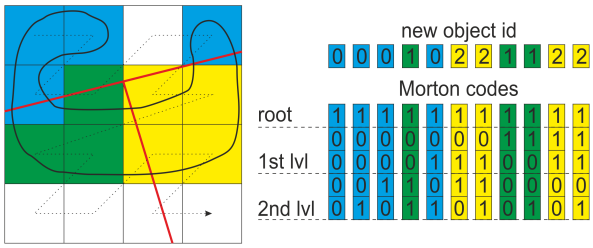


Figure 2: Fractured object represented by a quadtree. Node positions are described with Morton codes and new objects' ids are assigned.

but this would require a lot of additional memory space and extra processing during the SVO constructions. Instead, we decided to base our island detection algorithm on the direct application of the CCL underling conception realized in the form of a parallel implementation on GPU in CUDA.

In the initial step of the algorithm the voxels in FBS are labeled with consecutive numbers. Then, for each voxel within each VCP group one thread is launched. Every thread loops over the VCP voxels existing in the 26-neighborhood of the relevant central voxel, taking the maximum of the label values of the central voxel and the neighboring voxels and relabeling with the maximum the voxels of the lower label value. Since, at the same time, other "neighboring" threads may try to relabel the voxels, the relabeling is done using an atomic operation (`atomicMax` in CUDA). The central voxel's neighbors are determined on the basis of their Morton codes by means of the binary search on voxels from the same VCP group, and their memory addresses are cached in thread local memory. Each thread continues until there is no change in the labels of the voxels processed by the thread.

Then, for each VCP group in which relabeling took place, the threads are launched again, and the algorithm continues this way until there is no VCP group to relabel.

Finally, in the FBS voxels, the $\mho(V)$ indices are substituted with the labels, and FBS itself is stably sorted by the labels to obtain the required partition on the groups of the connected voxel subsets, which preserves the order of the Morton codes within each group.

4.5.2 Optimizations

The original 26-neighborhood each thread is supposed to operate on can be reduced by half. Let $\mathbb{V}$ be the voxel space induced by the cubic lattice that coincides with VCP; thus VCP $\subset \mathbb{V}$. Let $N_k(V)$ be a k-element subset of the 26-neighborhood $N_{26}(V) \subset \mathbb{V}$ of a voxel $V \in \mathbb{V}$. Assuming that the threads operate on the subsets $N_k(V) \cap$ VCP, $V \in$ VCP, it is easy to see that the sufficient condition for the correct propagation of labels by the threads within a connected component is that for all voxels $V, V' \in \mathbb{V}$ such that $V' \in N_{26}(V) \setminus N_k(V)$, the voxel V belongs to $N_k(V')$. If $N_k(V)$ satisfies this condition, then this guarantees that if $label(V) < label(V')$ for some voxel $V' \in N_{26}(V) \cap$ VCP, then there is a thread to propagate $label(V')$ to V. A little thought shows that for $k = 13$, $N_{13}(V)$ can be composed with 9 voxels of coordinates $(x+a, y+b, z-1)$ and 4 voxels of coordinates $(x+a, y+1, z)$, $(x-1, y, z)$, where $a, b \in \{-1, 0, 1\}$, and (x, y, z) are coordinates of the voxel V. Moreover, one can easily check that $k = 13$ is the minimum number of voxels for $N_k(V)$ to obey the above condition.

There are also other possible optimizations that we use in the island detection algorithm. First, there are dependency chains between voxels in the same neighborhood, which results in equivalence trees [SB10], that can be flatten in each iteration. Secondly, there is no need to re-launch threads which have propagated the VCP maximum labels to their neighborhoods. Since we label the FBS voxels, we know the maximum value of the labels within each VCP group, so we exclude such "maximum" threads from further operations. Thirdly, each voxel V stores the information about its *existing* neighbors (i.e., the voxels from $N_{13}(V) \cap$ VCP) in order to reduce the costly binary search being done by a thread only to these neighbors. The information is obtained during the first execution of the thread and coded in one integer by setting appropriate bits. Lastly, some speed/precision trade can be made, as the island detection can be performed on other SVO level than the finest one. In order to do so, one have to construct an approximation of FBS composed of the ancestor voxels of the FBS voxels at a given SVO level. The appropriate ancestors can be easily determined on the basis of the Morton codes of the FBS voxels. Then the island detection test can be carried out on that approximated FBS set, and the result propagated down to the voxels from the original FBS.

4.6 SVO Building

Having FBS partitioned on the connected voxel subsets distinguished by labels (Sec. 4.5.1), for each subset a SVO has to be built. As mentioned in Sec. 2, there are a number of fast methods for building a SVO, but we are interested on those that utilize Morton code. A SVO can be build in depth-first order [Kar12] by creating all levels in one kernel call. In turn, the breadth-first order is obtained iteratively [ZGHG11], and this approach is most suitable for our purposes. Moreover, the original algorithm can be easily extended to build many SVOs simultaneously (voxels from different SVOs are distinguished by the piece labels). Our implementation remains almost the same as in [ZGHG11], with two exceptions. First, the parallel prefix scan primitive is replaced with its segmented version [HB10]. Secondly, the sorting primitive operation has to be changed into

the stable sorting primitive and performed two times: first by Morton codes and second by the piece label.

4.7 Local Fracture

Apart from fracturing the whole object, there is also a possibility to perform this operation locally. In order to do that, one can attribute to every Voronoi seed an extra object identifier. Then, on the last level of applying the fracture pattern, voxels are assigned to piece objects on the basis of these identifiers instead of the seed indices (see Sec. 4.4). The assignment of the same identifier to many seeds allows large parts of the fractured object to remain intact and to properly create surfaces for pieces. However, the voxels intersected by Voronoi faces must still be detected, and on the last SVO level the intersection test must be altered. The modification concerns internal voxels intersected by a Voronoi face that is induced by seeds with the same identifier — these voxels should not be considered in the output result. Otherwise, some extra surfaces would be generated inside a part of the SVO that is supposed to stay intact.

5 SVO COLLISION AND RESPONSE

In this section we address the problem of the rigid body simulation of objects represented by SVOs. Lots of techniques have been proposed over the years to simulate physics behavior of objects in virtual environments. The methods have been mainly designed for meshes, but often tended to use other forms of representation and structures to model physical shapes and properties. For example, one may utilize particles located on the object's surface to detect collision and generate repulsion forces. However, the surface-based methods are characterized by short range of reactions and may not easily recover from deep penetrations. Another example are the distance-based methods. They rely on a spatial map that encodes the signed distance from a given point in 3D space to the object surface. Although they better handle deep intersections, the calculation of the distance map can be computationally expensive and memory consumptive.

The approach we chose to adapt to voxel domain is based on the volume minimization method [FBAF08]. In order to resolve collision, the technique models the intersection of colliding objects, and then computes the size of the intersection volume and the repulsion forces—the latter derived from the volume gradients. No precomputation is required and the efficiency of the surface-based methods is combined with the robustness of the distance-based ones.

5.1 Collision

In [FBAF08] to model the intersection of colliding objects the Layer Depth Images (LDIs) were used along with GPU to benefit from parallelism. Our approach is slightly different. First, the voxelization of objects into LDI is no longer necessary, since our objects are build with voxels. Secondly, in place of images of different resolutions, we take advantage of the SVO hierarchy and utilize it in the collision test.

5.1.1 Intersection of SVOs

In order to detect collision, we start with a broad-phase check in which the intersections of SVOs' roots of the objects are tested. Then, the potentially colliding objects are paired and their SVOs traversed simultaneously. When traversing down, pairs of voxels (a voxel from one SVO and a voxel from the other SVO) on a given level of the SVOs are examined for an intersection (performed between two AABBs in the local space of one of the SVOs), until the desired level of detail is reached. The process is very efficient, since the intersection test is continued only for the pairs of those child voxels whose parents intersect each other. Moreover, the pairs of internal voxels are omitted, since our goal is to follow the boundary voxels so as to enclose the intersection of the chosen levels, and calculate the size of the volume of the set and gradients of this volume. In turn, we use this information in order to generate the collision response forces (see 5.2).

5.1.2 Collision Points

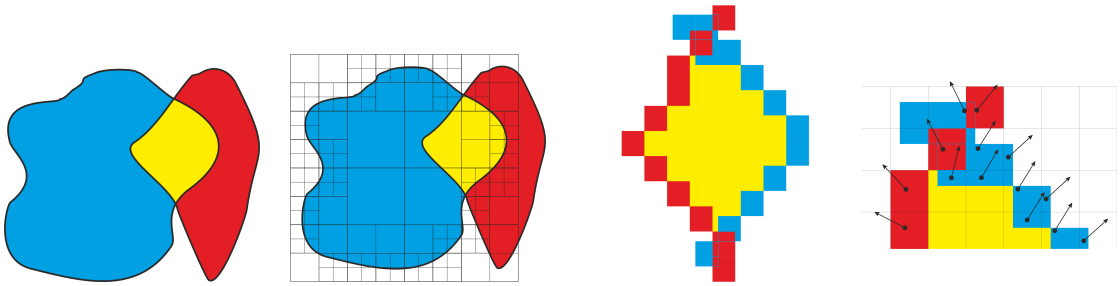
The next step is the approximation of the surface of the SVOs' intersection. For this purpose we generate a set of *collision points* when the desired SVO level is reached during the traversal of the trees. The collision points correspond to the intersections between the pairs of voxels from the colliding SVO objects. For a boundary-internal voxel intersection, the position of the collision point is determined as the center of the intersection volume between the pair of the voxels. In turn, for a boundary-boundary voxel intersection two collision points are generated; each is positioned on the face of one of the voxels from the pair by translating the center of the intersection volume along the normal vector of the voxel.

Moreover, with each collision point the normal vector of the surface of the SVOs' intersection is associated, which is just the normal vector of the corresponding boundary voxel.

Finally, we perform discretization of the collision points by averaging their positions and normal vectors within the cells of the cubic lattice coincident with the voxels on the chosen SVO level. The operation is straightforward as the collision points data is kept in the local space of one of the SVOs.

5.1.3 Intersection Volume and Gradients

Having the collision points, we can calculate the size of the intersection volume of the colliding objects. In order to do that, we integrate the intersection volume by



(a) (b) (c) (d)

Figure 3: (a) Two colliding objects. (b) The SVO structure drawn on the top of the blue object. (c) The intersection area modeled with voxels. (d) The dots represent averaged collision points and volume gradients.

iterating over the collision points located in the columns of the cubic lattice used for discretization. We examine pairs of subsequent collision points. Our goal is to find the pairs of the collision points such that their normal vectors' coordinates in the axis we iterate on have opposite signs, and the same requirement is fulfilled for the normal vector of the first collision point and the search direction. If this condition is satisfied, the pair of collision points represent the local entry and exit of the intersection volume surface, so the difference between the positions of such collision points (in the axis we iterate on), when multiplied by the voxel face area, is a partial volume of the intersection. Therefore, to obtain the entire volume of the intersection we sum these partial volumes. Moreover, we compute the volume gradients as the collision points' normal vectors multiplied by the voxel face area [FBAF08].

5.2 Response Forces

To resolve collision, the colliding objects have to be separated. For this purpose we utilize the penalty-based method as in [FBAF08]. It results in the generation of two forces for each collision point. First, there is a repulsion force, which can be formulated as:

$$f_{\text{rep}} = -kV_{\text{int}}g_i \tag{3}$$

where k is an arbitrary positive constant, V_{int} – the size of the intersection volume, and g_i – the volume gradient at the i-th collision point. The second force is the friction, expressed by the equation:

$$f_{\text{frc}} = \mu v_t a \tag{4}$$

where μ is an arbitrary positive constant, v_t – the relative tangential velocity, and a – the voxel face area.

6 RESULTS

In this section, we discuss the performance and quality of the presented solution. All depicted timings were obtained on Intel Core i7 960 CPU with Nvidia GeForce GTX Titan Black GPU. All algorithms were implemented using CUDA framework to fully exploit the parallelism delivered by GPU.

6.1 Varying Physics Level

Our collision detection method, due to the hierarchical structure of SVO, supports varying physics levels at no extra cost. As a result, one can easily balance between precision of performed calculations and performance. However, the object's volume is changing with physics level. The higher SVO level, the more accurate object's description. In order to obtain the volume of an object, its boundary voxels must be sorted along one axis and the pairs of the entry and exit voxels must be found. As a consequence, small objects at lower SVO levels may not have volume at all and collision forces would not have affect on them.

Moreover, the chosen physics level doesn't have to be the same for all collided objects' pairs. It can be adapted to the current situation or, for example, based on objects distance to the camera. When something happens far away from the observer, it doesn't have to be presented precisely, as the observer couldn't see all the details anyway. In addition, sorting collision points and forces calculations can be performed in parallel for pairs on different physics level.

6.2 Fracture Performance

Table 1 presents timings of consecutive fracture stages on chosen test scenes. Studying the results, one can observe that increasing SVO level with the same number of Voronoi seeds, multiplies the fracture time by a factor of 3–4 with reference to the fracture time on the previous SVO level. This is directly connected with the number of voxels on the current SVO level, which changes in a similar manner. Moreover, the impact location and the fracture pattern itself also has great influence on fracture time. The more fragmented objects, the more surfaces to create, and the more voxels to process.

6.3 Physics Performance

The physics part of our solution was tested using a number of various scenarios. First, the simplest scene is presented in fig. 4. The Bunny was fractured and new

Scene	SVO level	Voronoi nodes	FBS extraction	Voxels sorting	Islands detection	SVOs building	Internal nodes detection	Total
Bunny (Fig. 4)	8	90	2.5 ms	1.1 ms	2.7 ms	4.1 ms	2.5 ms	12.9 ms
	9	90	6.4 ms	2.5 ms	9.2 ms	5.9 ms	6.7 ms	30.7 ms
	10	90	21.0 ms	10.4 ms	40.1 ms	13.8 ms	25.5 ms	110.8 ms
Columns (Fig. 5)	10	70	17.5 ms	9.5 ms	38.8 ms	11.8 ms	22.2 ms	99.8 ms
Buddha (Fig. 6)	10	40	5.1 ms	3.4 ms	12.9 ms	6.3 ms	17.6 ms	45.3 ms
Dragons (Fig. 7)	9	15	2.1 ms	1.7 ms	6.8 ms	4.5 ms	7.0 ms	22.1 ms
		35	2.5 ms	2.1 ms	7.3 ms	4.9 ms	7.7 ms	24.5 ms
		50	2.7 ms	2.2 ms	8.5 ms	5.4 ms	8.8 ms	27.6 ms

Table 1: Fracture timings on different scenes. For the Columns and Buddha timings are an average of all fractures. In the Dragons scene, every dragon has an independent result, as different fracture parameters where used.

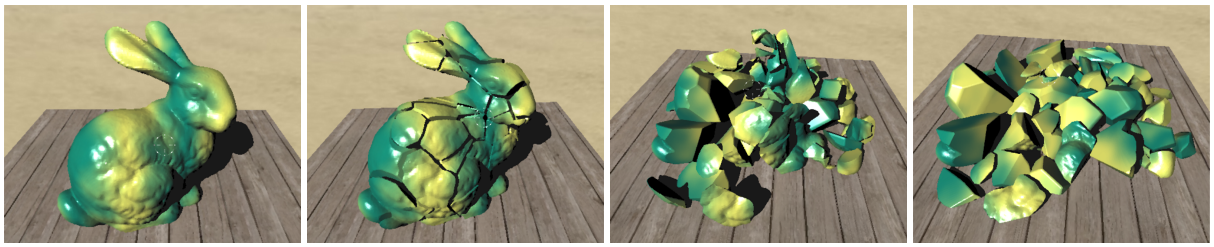


Figure 4: Fracturing the Stanford Bunny with the 8th physics level. Physics time was 4–9 ms per time step.

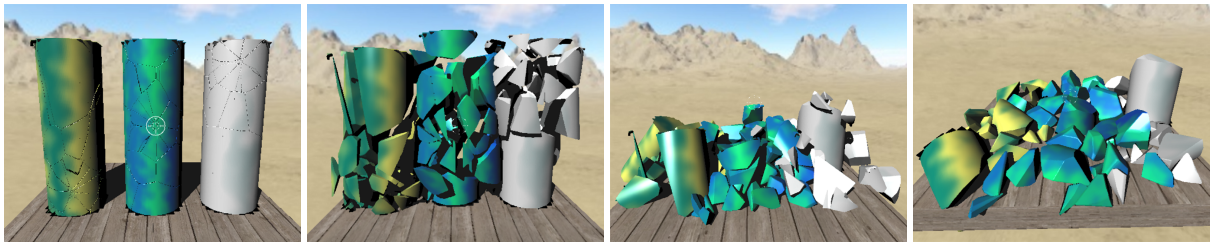


Figure 5: Fracturing three columns with physics calculated on the 7th SVO level. Physics time was 4–8 ms per time step.

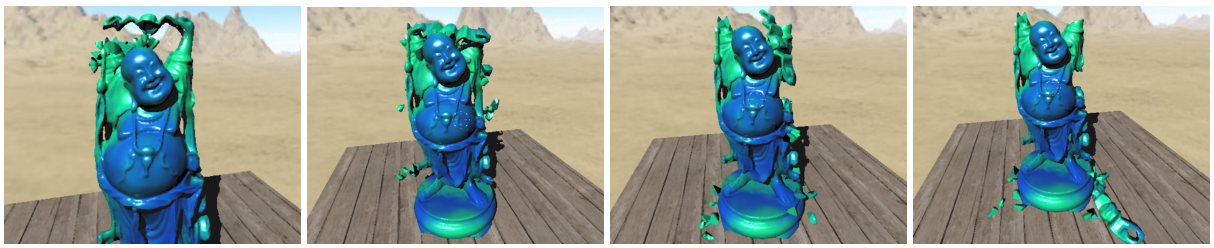


Figure 6: Local fracture of the Happy Buddha figure with the 9th physics level. Physics time was 5–7 ms per time step.

Figure 7: Three dragons dropped and fractured on collision points' locations. The objects had a different number of Voronoi seeds used during fracturing: 15, 35, 50. Physics was set to the 8th level and calculation time varied from 3 to 11 ms per time step.

pieces were scattered over the box's surface. The operation was rather energetic and rich with objects from a wide range of volumes. The physics time was 4–9 ms obtained on the 8th physics level. Even though, the lower physics level (7th) was used in *Columns* scene (fig. 5), the average time was similar to the previous one, and varied from 4 to 8 ms. It was the result of a larger number of collision points detected during the collision test. The opposite situation happened in *Happy Buddha* scene (fig. 6), where the 9th physics level produced better performance (5–7 ms) for local fracture. The last scene (fig. 7) presents three dragons dropped and fractured on collision points locations. The objects were cut into rather big pieces. However, due to their complexity, the average physics time varied from 3 to 11 ms. The time step for all simulations was set to 0.01 s, as we use the penalty-based response method with the implicit Euler method as an integrator.

7 CONCLUSION

We have presented a novel method for fracturing objects represented with sparse voxel octrees. Our method applies a fracture pattern to the object in the impact location, and cut the object into pieces, which are then also represented with SVOs. No precomputation of the pattern is required, as it is generated on-the-fly. After fracturing, new rigid objects are simulated. Thanks to our collision detection and response method, which is an extension of Faure's image-based approach [FBAF08], we can efficiently calculate physics in parallel on different levels of details with no extra cost.

8 REFERENCES

[AFC*10] Allard J., Faure F., Courtecuisse H., Falipou F., Duriez C., Kry P. G.: Volume contact constraints at arbitrary resolution. ACM Trans. Graph. 29, 4 (July 2010), pp. 82:1-82:10.

[Bau11] Bautembach D.: Animated sparse voxel octrees. Bachelor's Thesis (feb 2011).

[BCC*11] Baker M., Carlson M., Coumans E., Criswell B., Harada T., Knight P., Zafar N. B.: Destruction and dynamic artist tools for film and game production. In ACM SIGGRAPH 2011 course notes (2011).

[BLD14] Baert J., Lagae A., Dutra' P.: Out-of-core construction of sparse voxel octrees. Computer Graphics Forum 33, 6 (2014), 220-227.

[CNS*11] Crassin C., Neyret F., Sainz M., Green S., Eeisemann E.: Interactive indirect illumination using voxel cone tracing. Computer Graphics Forum (Proceedings of Pacific Graphics 2011) 30, 7 (sep 2011).

[Cra11] Crassin C.:. PhD thesis, Grenoble University, 2011.

[CYFW14] Chen Z., Yao M., Feng R., Wang H.: Physics-inspired adaptive fracture refinement. ACM Trans. Graph. 33, 4 (July 2014), 113:1-113:7.

[FBAF08] Faure F., Barbier S., Allard J., Falipou F.: Image-based Collision Detection and Response between Arbitrary Volume Objects. In Eurographics/SIGGRAPH Symposium on Computer Animation (2008).

[GLM96] Gottschalk S., Lin M. C., Manocha D.: OBBTree: A hierarchical structure for rapid interference detection, 1996.

[GPM11] Garanzha K., Pantaleoni J., Mcallister D.: Simpler and faster HLBVH with work queues. In Proceedings of the ACM SIGGRAPH Symposium on High Performance Graphics, HPG '11, ACM, pp. 59-64.

[HB10] Hoberock J., Bell N.: Thrust: A parallel template library, 2010. Version 1.7.0.

[HTG04] Heidelberger B., Teschner M., Gross M. H.: Detection of collisions and self-collisions using image-space techniques. In WSCG (2004), pp. 145-152.

[Hub95] Hubbard P. M.: Collision detection for interactive graphics applications. IEEE Transactions on Visualization and Computer Graphics 1 (1995), 218-230.

[IO09] Iben H. N.,O'Brien J. F.: Generating surface crack patterns. Graph. Models 71, 6 (Nov. 2009), 198-208.

[Kar12] Karras T.: Maximizing parallelism in the construction of BVHs, Octrees, and k-d Trees. In High Performance Graphics (2012), Eurographics Association, pp. 33-37.

[LK10] Laine S., Karras T.: Efficient sparse voxel octrees. In Proceedings of the 2010 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games, I3D '10, ACM, pp. 55-63.

[MCK13] Müller M., Chentanez N., Kim T.-Y.: Real time dynamic fracture with volumetric approximate convex decompositions. ACM Trans. Graph. 32, 4 (July 2013), 115:1-115:10.

[NTB*91] Norton A., Turk G., Bacon B., Gerth J., Sweeney P.: Animation of fracture by physical modeling. The Visual Computer 7, 4 (1991), 210-219.

[OBH02] O'Brien J. F., Bargteil A. W., Hodgins J. K.: Graphical modeling and animation of ductile fracture. In Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '02, ACM, pp. 291-294.

[OH99] O'Brien J. F., Hodgins J. K.: Graphical modeling and animation of brittle fracture. In Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '99, pp. 137-146.

[PK15] Pätzold M., Kolb A.: Grid-free Out-of-core Voxelization to Sparse Voxel Octrees on GPU. Proceedings of the 7th Conference on High-Performance Graphics, HPG '15, ACM, pp. 95-103.

[SO14] Schvartzman S. C., Otaduy M. A.: Fracture Animation Based on High-dimensional Voronoi Diagrams. Proceedings of the 18th Meeting of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games, I3D '14, ACM, pp. 15-22.

[SSF09] Su J., Schroeder C., Fedkiw R.: Energy stability and fracture for frame rate rigid body simulations. In Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation (2009), SCA '09, ACM, pp. 155-164.

[SB10] Stava O., Benes B.: Connected component labeling in CUDA. GPU computing gems emerald edition (2010), pp. 569-581.

[TF88] Terzopoulos D., Fleischer K.: Modeling inelastic deformation: Viscolelasticity, plasticity, fracture. SIGGRAPH Comput. Graph. 22, 4 (June 1988), 269-278.

[Wil13] Willcocks C. G.: Sparse volumetric deformation. PhD Thesis (apr 2013).

[WRK*10] Wicke M., Ritchie D., Klingner B. M., Burke S., Shewchuk J. R., O'Brien J. F.: Dynamic local remeshing for elastoplastic simulation. ACM Transactions on Graphics 29, 4 (July 2010), 49:1-11. Proceedings of ACM SIGGRAPH 2010, Los Angles, CA.

[ZBG15] Zhu Y., Bridson R., Greif C.: Simulating rigid body fracture with surface meshes. ACM Transactions on Graphics (to appear) (2015).

[ZGHG11] Zhou K., Gong M., Huang X., Guo B.: Data-parallel octrees for surface reconstruction. IEEE TRANSACTIONS ON VISUALIZATION AND COMPUTER GRAPHICS (2011).

Scene Understanding Using Context-based Conditional Random Field

Esfandiar Zolghadr
PhD candidate, CECS
Florida Atlantic University
777 Glades Road, Boca Raton, USA
ezolghad@fau.edu

Borko Furht
Professor, CECS
Florida Atlantic University
777 Glades Road, Boca Raton, USA
bfurht@fau.edu

ABSTRACT

In this paper, a new framework for scene understanding using multi-modal high-ordered context-model is introduced. Spatial and semantical interactions are considered as sources of context which are incorporated in the model using a single object-scene relevance measure that quantifies high-order object relations. This score is used to minimize semantical inconsistencies among objects in dense graph representation of the scene category during the object recognition process. New context model is later incorporated in a Conditional Random Fields (CRF) framework to combine contextual cues with appearance descriptors in order to increase object localization and class prediction accuracy. A novel context-based non-central hypergeometric unary potential is defined to maximize the semantical coherence in the scene. Further refinement is performed using context-based pairwise and high-order potentials which use alpha-expansion and graph-cut to find optimal configuration. Comparison between the purposed approach and state-of-art algorithms shows effectiveness of this approach in annotation and interpretation of scenes.

Keywords

Context-based scene recognition, supervised classification, generative model, representative feature

1. INTRODUCTION

Scene understanding has been studied for decades in areas such as content annotation, object and event recognition and media retrieval engines. The main objective is to provide more precise and accurate description of the scene in order to better respond to user queries. Media search engines currently use manually entered tags in metadata to recognize the content of images. Automated annotation frameworks are preferred methods for high volume contents. They utilize audio visual features to localize and classify candidate objects in an image which assigns class labels to components of a scene [1].

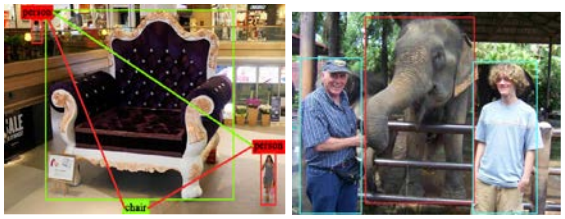
In a top-down approach, a global feature is extracted and used to classify the image as high-level categories such as (indoor/outdoor). Later, more specialized object detectors in that category are applied for detail analysis of the image. Torralba et al. [2, 3] presents a coarse scene-level global feature called 'gist' and use it to classify a scene as indoor or outdoor. This approach also allows checking for presence of an object types without running an object detector.

In a bottom up approach, individual objects detectors are applied to the extracted features from homogenous regions of the image to identify a matching object class. Key disadvantages of these paradigms are high dimensionality of the required detectors and region analysis in isolation. Previous work shows that performance of recognition systems could be significantly improved when scene-level knowledge known as "*context*" is exploited [4, 5, 6, 7, 8]. By definition, any information that can be used in accurate semantical recognition of scene elements and its underlying concepts is called context [9]. Contextual models can capture semantical properties, relationships and interactions among image components which can be used to infer higher level meaning of a scene.

Object detectors face many challenges due to poor image quality or overall image content complexity in cluttered images. Incorporating contextual information is used to disambiguate, refine and improve the recognition results. Additionally, context cueing can reduce dimensionality of required object detectors. Traditionally, contexts are constructed using semantical and spatial relations of objects in a scene [8]. More recent applications use additional types of contextual information provided by acquisition system such as sensory context (GPS) and 3D geometric scene context (orientations, support surface, horizon line) [9].

Semantic context is among most studied types of contexts which models object relationships such as co-occurrence statistics in an image. Spatial context captures inter-objects scale or location relationships based on various taxonomies such as horizontal or vertical relative positions [10].

a) Semantic consistency
Left: sofa outdoor (score=0.13),
Right: sofa in living room (score= 0.75)



b) Scale inconsistency
Left: extremely large chair (0.12)
Right: normal scaled elements (0.61)

c) Spatial Inconsistency
Left: car flying in the sky (0.29)
Right: car on the road (0.72)

Figure 1. These images show sample scenes and their contextual relevance scores obtained using higher-order relationship. Images on the right show objects consistent with scene context. Images on the left demonstrate contextual inconsistency (higher scores signifies more consistency).

The previous work on context-based detectors is mostly limited to studying the objects in pairwise relations with less attention to higher order relations and structure of scene layouts. An image of real life scene configuration can be rather complex where multiple types of contexts should be considered to explain high-ordered relationships.

In this paper, a novel multi-source high-order contextual scene recognition framework is introduced to represent realistic scene configurations. This framework measures contextual consistency among the composing elements of an image using a measurement called "*object-to-scene relevance score*" which measures contribution of an object type to overall semantical meaning of the scene for a given context. The object relevance score is used in modeling underlying scene representation based on high-ordered relationships in the form of undirected graph.

In summary contributions of this work is as follows:

- Definition of a context-based conditional random field designed to incorporate multiple source of context in high-order relationship.
- Definition of object-scene relevance score that encodes layout, relations and interactions of an object to conform to scene consistent context.
- Use of a novel unary potential based on non-central hypergeometric distribution to predict the object labels in a context-based generative process.
- We define a high-order potential to encode high-order contextual relationship of the objects.

The rest of this paper is organized as follows. Section II is an overview of the related work. In Section III we present our framework in detail. Section IV presents our experimental results. Section V is concluding remarks.

2. RELATED WORK

Contextual scene understanding frameworks have been studied in many of the previous work [11, 12]. Wolf and Bileschi [13] introduced "semantic layers" which are constructed by extracting and combining various features such as color, texture, geometric feature maps and saliency maps at pixel location during the learning stage. Each semantic layer represents an object category indicate the presence of a particular object in the image at a semantic layer.

Galleguillos *et al.* [14] explored pairwise interactions between pixels, regions and objects to extract and learn three source of context semantic, boundary support and contextual neighborhood.

Torralba *et al.* [3] introduced a simple framework for modeling the relationship between context and object properties. Scale context was used to provide a strong cue for scale selection in the detection of high-level structures as objects. Contextual features were obtained from a set of training images and object properties were based on the correlation between the statistics of low-level features across the entire scene.

Choi *et al.*[15] used tree-structure graphical model to encode hierarchical dependencies among object categories and scenes. They used contextual score to quantify pairwise information such as position and scale relationship.

Jones and Shao [16] studied pairwise contextual interactions of events and scene elements in a clustering application. They demonstrated performance improvement over state-of-the-art clustering methods.

High-order relationships are examined in [17, 18, 19] on single source of context such as co-occurrence. The shortcoming of these methods is when discriminative contextual cues may appear in other contextual modalities such as scale or spatial context as is illustrated in Figure 1.

On the other side of spectrum, generative models such as [20] are widely used to model multi-context relations. The limitation of these frameworks and generally the generative process is the independence assumption on

observed data to make the inference tractable which is very restrictive.

Previous work shows success of context-based methods in improving performance of object localization and recognition. We extend previous work to exploit high-order multimodal contextual relationships instead of pair-wise approach. We propose a high-order context framework that learns appearance, structure and semantical consistency of the scene and infers its parameters based on multi-modal context sources for domain object types. Objects co-occurrence statistics is defined in high-order to capture scene level semantics. For example objects in {car, motorcycle, road, sky} tend to appear in outdoor street images and {car, truck, rubber duck, Mickey Mouse} represents set of children toys.

Spatial and scale contexts are sources of layout topology. Location and scale information is obtained from bounding box information in training dataset and transformed into the set of contextual spatial attributes during learning process. Bounding box information is acquired from the image annotations provided in Sun397[1] dataset. This dataset provides a better alternative to Google Sets or web documents used in some related works [21].

Context model represents a scene with fully connected graph consisting objects at each node. These nodes are connected with undirected edges. Each edge is assigned with contextual relevance measurement that quantifies the relations between two objects given the dominated context in that clique. As shown in Figure 1, contextual relevance is defined to maximize semantical consistencies including scale and location in a scene. Contextual inconsistencies may not manifest in pairwise relations where in ternary relation a clear violation is evident. The object-scene score is scalable and extendible to other datasets since it not dependent on visual primitives. Contextually related objects form semantically coherent cliques in our graph representation and are labeled according.

Conditional random fields (CRF) [22], a discriminative framework is used to incorporate contextual cues along with appearance features in a single model. This allows to model intrinsic and extrinsic structure of an image for better understanding of its underlying concepts [23]. Given observed variable X, CRFs model the conditional distribution of Y given X to encode complex dependencies of Y on X. In this paper definition of CRF is extended by conditioning on visual features and context which is called Context-based CRF (CBCRF). CBCRF combines appearance descriptors, contextual relations and layout structure of the objects likely to be present in that scene category.

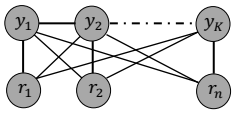
Our experiments show that contextual relations of high-order can improve object detection, scene classification and can be used in many other applications such as detection of out-of-context or black-boxed object.

[1] http://vision.princeton.edu/projects/2010/SUN/

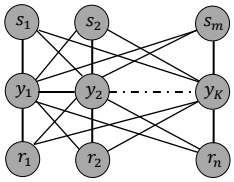
3. MODEL AND ALGORITHMS

A conditional random field (CRF) model [22] is used to learn the conditional distribution over the set of class labels given an image. The following is formalization of the model:

With K being total objects in our dataset, let's consider a random field Y defined over set of variables $\{y_1, \cdots, y_K\}$ to represent labels of all objects. Domain of each variable y_i is $\mathcal{L} = \{l_1, \cdots, l_K\}$ which is set of all possible labels. Let $X = \{x_1, \dots, x_D\}$ be the set of images in our dataset, $R_i = \{r_1, \cdots, r_n\}$ be set of visual words of i^{th} image, $C_i = \{c_1, \cdots, c_K\}$ be image sub-region category labels representing objects, and $S_i = \{s_1, \cdots, s_K\}$ be set of contexts under which independent measurement of semantic relevance calculated for detected objects in i^{th} image. Each image is composition of arbitrary number of object instances in the same scene category.



a) Fully Connected CRF



b) Context-based CRF

Figure 2. Graphical representation of the CRF model (top) and context-based CRF model (below)

The use of a conditional random field allows us to incorporate appearance based descriptors, layout, and location cues in a single unified model. Our context-based CRF approach aims to find optimal configuration $Y = \{y_1, y_2, \dots, y_n\}$ which is characterized by Gibbs distribution $P(Y|R)$:

$$P(Y|R,\theta) = P(Y|\, R,S,\theta)\; P(S|R,\theta)$$

where θ is model parameters, S is context and $E(Y|S,R,\theta)$ is the probability of the labeling configuration Y given visual words conditioned on the context the conditional random field defined as:

$$P(Y|\, R,S,\theta) = \frac{1}{Z(Y,S)} \exp\big(-E(Y|R,S)\big) \qquad (1)$$

where Z is normalization partition function.

Our model is fully connected CRF with unary, pairwise and high-order potentials with following Gibbs Energy:

$$E(Y|\, R,S) = \sum_{n\in N} \psi_u(y_n) + \sum_{(i,j)\in P} \psi_p(y_i, y_j) + \sum_{i\in H} \psi_h(y_i) \qquad (2)$$

where N, P, H are number of candidate objects in the image, number of pairwise and high-order cliques respectively.

Scene Relevance Score

Context-based conditional random field model builds on the "*Scene Relevance Score*" (SRS) which is calculated using high-ordered interaction of the objects in each scene category.

The high-order pure independence rule [24] is used to define spatial context probabilities. Let $C = \{c_1, c_2, \dots, c_n\}$ be set of object types in our dataset, then A^K represents the set of possible combinations of object types with *K* object present and $n - K$ not present. For example considering set of $C = \{c_1, \dots, c_4\}$ with four object classes, the set of object configurations with only two objects present could be expressed as $A^2 = \{0011, 0101, 0110, 1010, 1100\}$.

Scene relevance-score is defined as posterior probability which is log likelihood of spatial, scale and semantic contexts:

$$\tau_{1\dots n} = log \prod_{k=0}^{n} \prod_{a \in A^K} \left(P_{L^v_{1\dots n}|a}\right)^{(-1)^{n-k}} \left(P_{X_{1\dots n}|\mathrm{a}}\right)^{(-1)^{n-k}} \quad (3)$$

where $P_{L^v_{1\dots n}|a}$ is spatial context and is defined as posterior probability of vertical location of an object in respect to others in a high-order relation. $L^v_{1\dots n}$ is high order relative vertical location configuration defined as joint probability distribution of $L_1, L_2, \dots, L_n$. $L_i \in \{above, below, even\}$ and is determined be comparing centroids of each object's bounding box. For example expected relative location of "*Sky*" is "*above*" the object "*Grass*".

$P_{X_{1\dots n}|\mathrm{a}}$ is high-ordered scale context and is defined as joint probability distribution of $X_1, X_2, \dots, X_\mathrm{n}$ where X_i is the expected relative scale relation obtained by transforming the image plane into 3D coordinates for relatives scale measurements based on labeled training sets.

Information obtained from relative horizontal location does not offer discriminative information and is not modeled.

The semantical relationship is implicitly encoded in scene relevance score in Equation (3) which shows strong semantical correlation for positive values and negative correlation for negative values of $\tau_{1\dots n}$ and zero for no relation.

Normalizing τ transforms the value to zero and one range which more suitable to our context model. The following function transforms the τ to normalized form:

$$\overline{\tau_{1\dots n}} = \frac{1}{1 + exp(-\tau_{1\dots n})}$$

The normalized value of $\overline{\tau_{1\dots n}}$ is interpreted as follows:

$$\begin{cases} 0.5 < \bar{\tau} \leq 1 & \text{semantical related} \\ \bar{\tau} = 0.5 & \text{no relation} \\ 0 \leq \bar{\tau} < 0.5 & \text{negative relation} \end{cases} \quad (4)$$

Strength of relationship increases with the value of $\bar{\tau}$ from 0 (impossible) up to 1 (strongly coupled).

Unary potential

The model appearance, affinity and shape are modeled using unary potential ψ_u. Unary potential is the most important potential and is sensitive to mislabeling as a result of initialization. By incorporating context the classification of objects is influenced by dominant context and hence initially misclassified labels can be refined. Unary potential is defined as:

$$\psi_u(y_n) = p(Y|R, S) \quad (5)$$

where $S_i = \{s_1, \cdots, s_m\}$ is all possible context graphs and the term $p(Y|R, S)$ is probability that object i^{th} would be assigned label *y* given the relevance score of the object.

Wallenius Latent Dirichlet Allocation (WLDA) [28] is a generative process for object localization. An image is partitioned into related groups of visual words which represent candidate objects and assigns best annotation label to the image category. In this process each label is associated with image feature data as response variable which is influenced by contextual constraints as bias weight parameter in Wallenius distribution.

The generative process of annotating a candidate object with its class label response variables is as follows:

- Draw topic proportions from Dirichlet prior $\theta \sim Dir(\alpha)$.
- For each visual word $R_n, n \in \{1, 2, \dots, N\}$:
  - Draw topic assignment $z_n|\theta \sim Mult(\theta)$
  - Draw region visual word $r_n|z_n \sim Mult(\beta_r)$
- For each object class label
  - draw a Wallenius c_i conditioned on contextual constraints given by $p(Y|Z, S) \sim Wall(Y, Z, S)$

where Y is response variable, Z is a set of topics equivalent to candidate objects, and S is context.

The objective is to obtain probability of most semantically consistent labeling configuration Y given topic distribution:

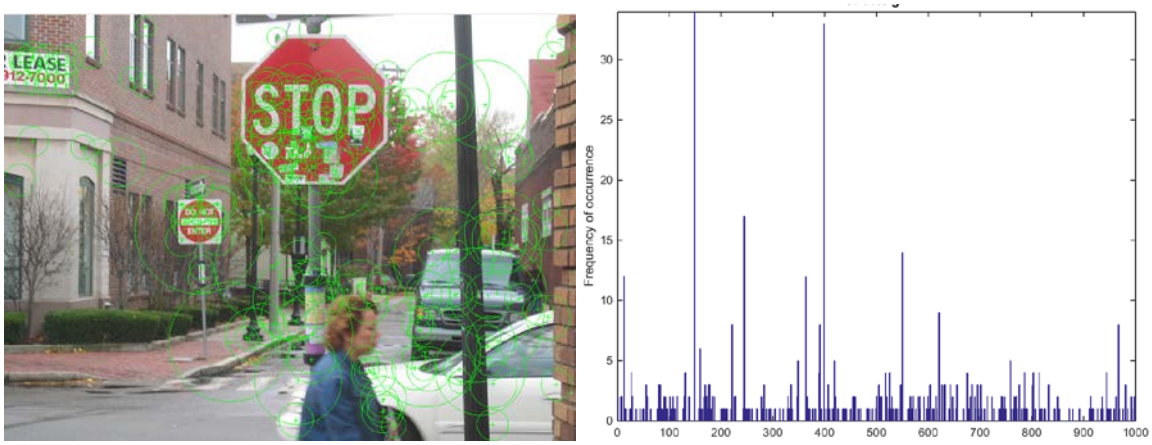
$$\begin{aligned} p(Y|Z, S) &= \Lambda(Y, Z) I(Y, Z, S) \\ \Lambda(Y, Z) &= \prod_{i=1}^{k} \binom{y_i}{z_i} \\ I(Y, Z, S) &= \int_0^1 \prod_{i=1}^{M} \left(1 - t^{\frac{s_i}{\sum_{i=1}^{M} s_i}}\right)^{y_i} dt \end{aligned} \quad (6)$$

Computing integral in Equation (6) is intractable because of the fractional exponent and must be approximated. First we simplify the formula for binary variables ($\Lambda(Y, Z) = 1$) as follows:

$$I(Y, Z, S) = \int_0^1 \prod_{j=1}^{M} \left(1 - t^{\frac{s_j}{s}}\right) dt \quad (7)$$

where $s = \sum_{i=1}^{M} s_i$.

Integrand in Equation (8) can be transformed to an easier to solve polynomial using variable substitution described in Equation (8).



a) Original image **b) Encoded image visual words histogram for the image shown in (a)**

Figure 3- Encoding of a sample image in corresponding visual word frequencies.

The polynomial of Equation (8) can be easily solved by scaling the context values to integer and reducing them by dividing to the greatest denominator.

$$\begin{aligned} \mathrm{t} &= \mathrm{u}^{s} \\ dt &= s.\,\mathrm{u}^{s-1}du \\ p(Y|Z,S) &= \int_0^1 \prod_{j=1}^{M}\left(1 - t^{\frac{s_j}{s}}\right)dt \\ &= \int_0^1 \prod_{j=1}^{M}\left(1 - (\mathrm{u}^{s})^{\frac{s_j}{s}}\right).s.\,\mathrm{u}^{s-1}du \\ &= s.\int_0^1 \mathrm{u}^{s-1}.\prod_{j=1}^{M}(1-u^{s_j})dt \end{aligned} \tag{8}$$

Pairwise potential

The pairwise term $\psi_p(y_i, y_j)$ reinforces contextual compatibility between label assignments of the neighboring object. It predicates on the assumptions that objects (or pixels) adjacent to each other are more likely to have the same label or be semantically related. Probability of label assignment follows the given context. This potential takes the form of Potts model to penalize semantically incompatible labels:

$$\psi_p(y_i, y_j) = \begin{cases} 0 & if\ y_i = y_j \\ \lambda_p exp\left(-|l_i - l_j|^2\right) & otherwsie \end{cases} \tag{9}$$

where $l_n = p(y_n|S)$ and λ_p is parameter whose value is learned from training data. This potential has shrinkage bias which means it only enforces label consistency in adjacent objects.

High-ordered potential

The high-order potential $\psi_h(y_i)$ is defined to maximize contextual consistency and compatibility of the label assignment in neighborhood of an object. To achieve this, objects in an image are grouped in semantically compatible and consistent cliques [19]. A penalty is applied to non-relevant ones to disassociate them from clique. Consistency of the clique is measured using the variance of unary feature response evaluated on all objects in that clique as follows:

$$\vartheta_C = exp\left(-\frac{\|\sum_{c\in C}(I_c - \mu)^2\|}{|C_L|}\right)$$

Where C_L is the clique, $|C_L|$ is cardinality of that clique, $I_c = p(y_n|C)$ and $\mu = \sum_{n\in C} p(y_n|C)/|C_L|$. Given the CRF model in Equation (2), high-order potential is defined as following:

$$\psi_h(y_i) = \begin{cases} N\lambda_h\vartheta_C\dfrac{1}{Q} & if\ N \le Q \\ \lambda_h\vartheta_C & otherwise \end{cases} \tag{10}$$

where N is number of elements in the clique y_i with label assignment that are inconsistent with dominant label in that clique and λ_h is model parameter which is obtained during the training. Consistency of this potential is controlled by threshold parameter Q which defines a cut-off point where from that point stronger penalty is imposed on very semantically consistent cliques. With the objective of finding the most probable labeling configuration that maximizes the conditional probability of Equation (1), alpha-expansion graph-cut optimization algorithm [25] is applied to get the optimal configuration $y^* = [y_1^*, y_2^*, \dots, y_n^*]^T$.

$$y^* = arg\,max\,P(y|R) = arg\,min\,E(y) \tag{11}$$

where y_n^* is unit basis vector that represents the result of object localization for n^{th} object in the image.

Contextual relevance is used during the optimization to eliminate false positives and keep correct detections.

4. EXPERIMENTS

Dataset

Object recognition algorithm was evaluated on subset of SUN397 datasets with 2152 images randomly selected as training set and 2010 images selected as test set from 62 object categories. The metadata of labeled images were used to extract images of objects according to their bounding box information. In pre-processing phase, images were scaled to meet a minimum dimensional constraint.

Training

Image feature space was represented as Bag-of-Features (*BoF*)[26]. Each code-word in the dictionary is a visual

appearance feature which was constructed based on "Speeded Up Robust Features" (SURF) [27] algorithm. SURF feature points were obtained from 64x64 blocks for image objects and transformed into descriptors. Top m strongest SURF descriptors were selected and normalized across entire training set. The value of m is calibrated empirically. Selected descriptors were then quantized into vocabulary sizes of V visual words using K-means clustering algorithm. Figure 3-(b) illustrates BoF representation of an image in (Figure 3-(a)) encoded as histograms of visual words (V =1000) which is used to train our model.

There are two sources of parameters in this study. The first one is the LDA parameter set which is learned the way is described in [28]. The second set of parameters is the CRF parameter set $\{\lambda_p, \lambda_h\}$. These parameters were all learned from the training set using the same method introduced in [19].

Evaluation Methods

For evaluation of context-based CRF framework, multiclass support vector machine (SVM) [29] classification method was used as baseline and compared to the state-of-the-art tree-based contextual model [15] using code provided at their site.

Metrics

Normalized mutual information (NMI) [30] is a metric used to evaluate performance of clustering and to measure how well objects in test images are assigned to object categories. NMI is a number between 0 and 1 and with 1 being perfect object label assignment and is calculated as follows:

$$NMI = \frac{\sum_{h,l}|x_{h,l}|\, log\left(\frac{|X|.|x_{h,l}|}{|x_h|.c_l}\right)}{\sqrt{\left(\sum_h |x_h|\, log\left(\frac{|x_{h,l}|}{|X|}\right)\right)\sum_l log\left(\frac{c_l}{|X|}\right)}} \quad (12)$$

where X is set of images, x_h is set of images in class h, $x_{h,l}$ is number of images that are member of both classes h and l and c_l is images labeled as class l.

Figure 4 illustrates object detection *NMI* that was applied to the models in these experiments. The results show context-based CRF model performs better in various topic sizes of K. These experiments also demonstrate that larger number of the topics have little impact on the object detection performance but has serious computational cost and performance degradation as the number of topics increases. When a scene contains less than K objects, the absent object categories will have very few or no members such that the impact will be small enough to be neglected. The optimum value of K is determined empirically and set to 150.

To evaluate performances of our framework for localization and presence F1-Measure was used which is a balanced score between precision and recall (F1) as follows:

$$F1 = \frac{2 \times Precesion \times Recall}{Precesion + Recall} \quad (13)$$

Classification performance was evaluated using objects labels in Ground-truth.

Figure 5 shows how this mettric was used in finding optimum parameter values for pairwise and high-order potentials.

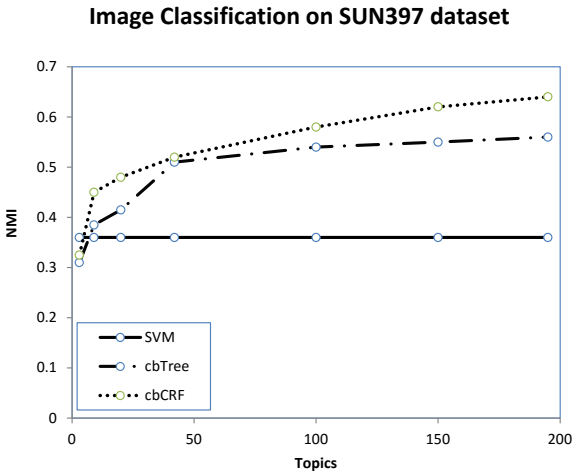


Figure 4 Object detection performance using NMI metric (1.0 is most accurate)

Model Parameters

Parameters that have influence on the distribution of topics in potentials were also investigated. There are two main parameters that require calibration, pairwise (λ_p) and high-order parameter (λ_h).

The tuning result on SUN397 is given in the top chart of Figure 5. Parameters λ_p and λ_h varied independently from 0 to 1 with interval 0.1 to pick the optimum value. As is illustrated in Figure 5, the performance improves as the value of the parameters increases. Slightly sharper gain in high-order potential than pair-wise demonstrates effectiveness of this potential.

Result of Empirical Study

To build the framework, a graph was constructed for each scene type to maximize contextual consistency. First scale and location context scores were calculated for all object pairs ($\overline{\tau_{ij}}$) in that image using Equation (4). Pairwise relations with $\overline{\tau_{ij}} > 0.5$ were added to the graph and others were ignored. Next, high-order context for all cliques combinations (i.e. $\overline{\tau_{1..k}}$) were computed and the clique with highest average score was selected as dominance context. The context model was fitted using Gaussian distribution for each context type which later was used in building CRF model to predict correct label assignment for candidate objects.

Table 2 shows the comparisons between baseline detector, SVM, tree-based context and our framework. From the table, we see our framework produces the best performance in both object localization and presence.

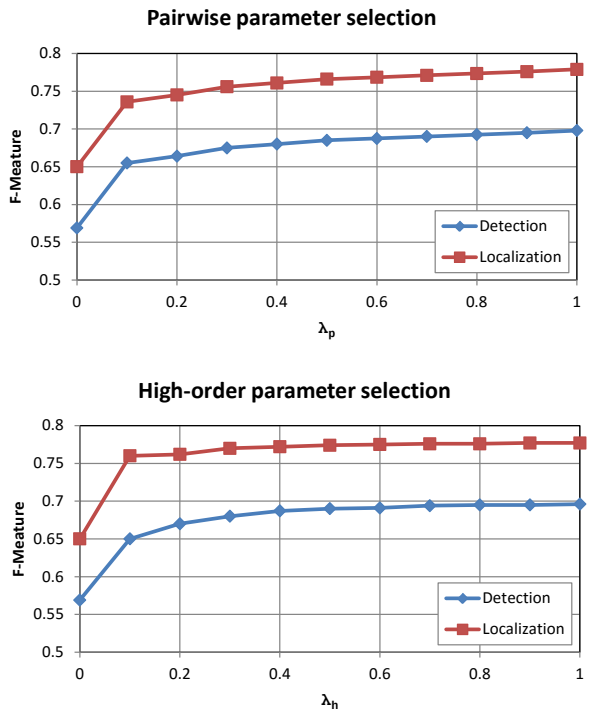


Figure 5. Parameter selection for pairwise and high-order potentials.

As shown in Table 1, the localization improvement over baseline detector algorithm is about 37.2% and the improvement over the state-of-the-art context model (tree context) is 7.5%. For the presence, the corresponding improvements are 37.6%, 14% respectively.

Metric	Localization	Presence
SVM	50.2	57.7
Tree-Based	64.1	69.6
Context-based CRF	**68.9**	**79.4**

Table 1. Object localization and presence performance comparison

Performances of proposed framework for object detection is illustrated in Figure 6 which shows improvement over the tree-context model for most object categories.

Table 2 shows some examples of results in which context constraints are strictly enforced to facilitate the contextually consistent detections.

Results shown illustrates that context-based CRF has improved compare to performance of the SVM and CRF in classification of the objects.

5. CONCLUSIONS

In this paper, we presented a discriminative model that combined the power of a generative model as unary potential and used an object-scene relevance score to encode pair-wise and high-order semantic contexts. We showed how to encode the high-order relationship among objects and build a robust models to enforce location, scale and semantical constrains. We compared our framework with other context-based model which employed similar sources of contexts in pairwise relations.

	SVM	Tree Context	Context-based CRF
Bed	0.53	0.64	0.71
Bicycle	0.59	0.72	0.78
Cabinet	0.44	0.53	0.54
Car	0.58	0.80	0.88
Keyboard	0.52	0.64	0.72
Monitor	0.46	0.63	0.66
Street sign	0.52	0.68	0.72
Table	0.37	0.49	0.50

Table 2- Object detection performance comparison

Our results demonstrated that our framework outperformed the current state-of-the-art context-based object localization methods. Our generative process implemented a true context-based approach where the context was directly applied to classification problem as unary potential. We showed an inference method to solve

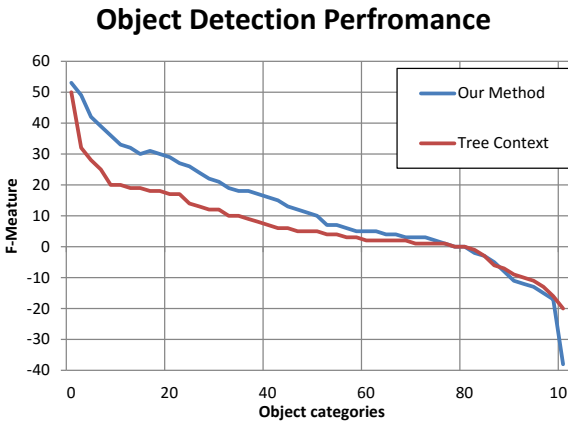
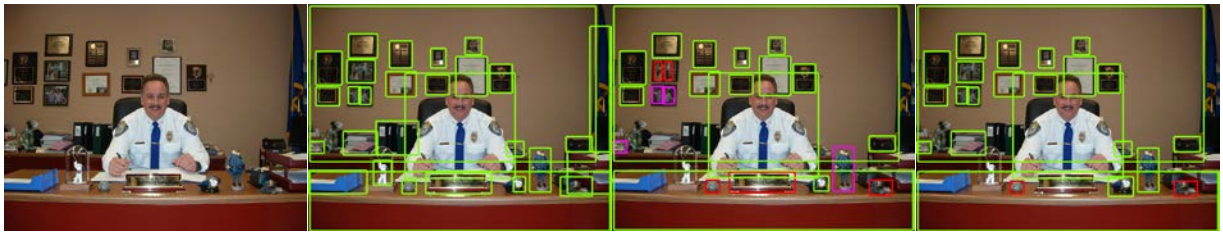


Figure 6. Object detection performance of CBCRF method compare to tree context method using SUN397 dataset.

the intractability problem of the WLDA to a solution that could be solved at polynomial time. We then applied our framework to distinguish the contextual consistency of the candidate objects using various contextual cues.

During our experiments we observed two main weaknesses. First, building a meaningful contextual relevance score requires presence of large number of objects in a scene category with ternary or more interactions. This is limiting factor that restricts choice of training dataset. Second drawback is relatively high computation requirement of this method, which is a side effect of WLDA generative process.



(a) Original image **(b) Ground Truth** **(c) Baseline Detector** **(d) CBCRF**

Figure 7- Sample object detection result. Green bounding boxes are correctly classified labels.

Our results demonstrated that use of context-based high-ordered potentials has outperformance advantages over the base-line and the state-of-the-art context based object detectors.

6. REFERENCES

[1] Wang C., Blei D. , Fei-Fei L. (2009), Simultaneous Image Classification and Annotation Computer Vision and Pattern Recognition, 2009. Paper presented at the ISSN: 1063-6919 , IEEE CVPR 2009. IEEE Conference on 20-25 June 2009 10.1109/CVPR.2009.5206800

[2] Torralba, A., Contextual Priming for Object Detection, Int'l J. Computer Vision, vol. 53, pp. 169-191, 2003.

[3] Murphy, K.P. Torralba, A. and Freeman, W.T., Using the Forest to See the Trees: A Graphical Model Relating Features, Objects and Scenes, Proc. Neural Information Processing Systems, 2003.

[4] Wang, J., Chen, Z., & Wu, Y. (2011). Action recognition with multiscale spatio-temporal contexts. Paper presented at the 3185-3192. doi:10.1109/CVPR.2011.5995493

[5] Choi, M. J., Torralba, A., & Willsky, A. S. (2012). Context models and out-of-context objects. Pattern Recognition Letters, 33(7), 853-862. doi:10.1016/j.patrec.2011.12.004

[6] Zhu, Y., Nayak, N. M., & Roy-Chowdhury, A. K. (2013). Context-aware modeling and recognition of activities in video. Paper presented at the 2491-2498. doi:10.1109/CVPR.2013.322

[7] Zhang, L., Kalashnikov, D. V., Mehrotra, S., & Vaisenberg, R. (2014). Context-based person identification framework for smart video surveillance. Machine Vision and Applications, 25(7), 1711-1725. doi:10.1007/s00138-013-0535-8

[8] Galleguillos, G. and Belongie, S., Context based object categorization: A critical survey, Comput. Vis. Image Underst., vol. 114, no. 6, pp. 712–722, Jun. 2010.

[9] Marques, O., Barenholtz, E., Charvillat, V.. Context modeling in computer vision: techniques, implications, and applications, Multimedia Tools and Applications, 2011) 51:303–339

[10] Fink M, Perona P (2003), Mutual boosting for contextual inference. In: Thrun S, Saul L, Schökopf B (eds) Advances in neural information processing systems (NIPS). MIT Press, Cambridge, MA

[11] Tang, J Shao, L., and Zhen, X., Robust point pattern matching based on spectral context, Pattern Recognit., vol. 47, no. 3, pp. 1469–1484, 2014.

[12] Jones, S. and Shao, L., Unsupervised spectral dual assignment clustering of human actions in context, in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Columbus, OH, USA, 2014, pp. 1–8.

[13] Wolf, L., & Bileschi, S. (2006). A critical view of context. International Journal of Computer Vision, 69(2), 251-261. doi:10.1007/s11263-006-7538-0

[14] Galleguillos, C., McFee, B., Belongie, S., and Lanckriet, G., Multi-class object localization by combining local contextual interactions, in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), San Francisco, CA,USA, Jun. 2010, pp. 113–120.

[15] Choi, M. J., Torralba, A., & Willsky, A. S. (2012; 2011). A tree-based context model for object recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence, 35(2), 240-252. doi:10.1109/TPAMI.2011.119

[16] Shao. L., Jones. S. , and Xuelong, L., Efficient search and localization of human actions in video databases, IEEE Trans. Circuits Syst. Video Technol., vol. 24, no. 3, pp. 504–512, Mar. 2014.

[17] Chen, G., Ding, Y., Xiao, J., and Han, T., Detection evolution with multi-order contextual co-occurrence, in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Portland, OR, USA, Jun. 2013,pp. 1798–1805.

[18] Myeong, H. & Lee, K. M., Tensor-based high-order semantic relationtransfer for semantic scene segmentation, in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Portland, OR, USA, Jun. 2013,pp. 3073–3080.

[19] Kohli, P. & Torr, P. H., Robust higher order potentials for enforcing label consistency, Int. J. Comput. Vis., vol. 82, no. 3, pp. 302–324,2009.

[20] Fergus, R., Perona, P., & Zisserman, A., Object class recognition by unsupervised scale-invariant learning. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, volume 2, pages 264-271, June 2003.

[21] Bengio, S., Dean, J., Erhan, D., Ie, E., Le, Q., Rabinovich, A., Singer, Y. (2013). Using web co-occurrence statistics for improving image categorization.

[22] Lafferty, J., McCallum, A. and Pereira F.. Conditional random fields: probabilistic models for segmenting and labeling sequence data. In Proceedings of the 18th International Conference on Machine Learning, 2001.

[23] Torralba, A., Murphy, K.P., and Freeman, W.T., Contextual Models for Object Detection Using Boosted Random Fields, Advances in Neural Information Processing Systems, MIT Press, 2005.

[24] Hou, Y., He, L. Zhao, X., and Song, D., Pure high-order word dependence mining via information geometry, in Proc. Adv. Inf. Retrieval Theory, Bertinoro, Italy, 2011, pp. 64–76

[25] Boykov, Y. and Veksler, O., Graph cuts in vision and graphics: Theories and applications, in Handbook of Mathematical Models in Computer Vision. New York, NY, USA: Springer, 2006, pp. 79–96.

[26] Csurka, G., Dance, C. R., Fan, L., Willamowski, J., and Bray, C., Visual categorization with bags of keypoints, in Proc. ECCV Workshop Stat. Learn. Comput. Vis., 2004, pp. 1–22.

[27] Bay, H., Tuytelaars, T., and Van Gool, L.. Surf: Speeded up robust features. In Computer Vision–ECCV 2006, pages 404–417. Springer, 2006.

[28] Zolghadr, E., & Furht, B. (2016). Context-Based Scene Understanding. International Journal of Multimedia Data Engineering and Management (IJMDEM), 7(1), 22-40. doi:10.4018/IJMDEM.2016010102

[29] Desai, C., Ramanan, D., and Fowlkes, C. C. Discriminative models for multi-class object layout, Int. J. Comput. Vis., vol. 95, no. 1, pp. 1–12, 2011

[30] Strehl, A., Ghosh, J., and Mooney, R. Impact of similarity measures on Web-page clustering, in Proc. the Workshop on Artificial Intelligence for Web Search, pp. 58-64, Austin: AAAI Press, 2000.

Multiresolution Laplacian sparse coding technique for image representation

Intidar Jemel
REsearch Laboratory in Intelligent Machines
University of Sfax, ENIS
BP 1173, Sfax, 3038, Sfax, Tunisia
intidhar_jemel@yahoo.fr

Ridha Ejbali
REsearch Laboratory in Intelligent Machines
University of Sfax, ENIS
BP 1173, Sfax, 3038, Sfax, Tunisia
ridha_ejbali@ieee.org

Mourad Zaied
REsearch Laboratory in Intelligent Machines
University of Sfax, ENIS
BP 1173, Sfax, 3038, Sfax, Tunisia
mourad.zaied@ieee.org

ABSTRACT

Sparse coding techniques have given good results in different domains especially in feature quantization and image representation. However, the major weakness of those techniques is their inability to represent the similarity between features. This limitation is due to the separate representation of features. Although the Laplacian sparse coding doesn't focus on the spatial similarity in the image space, it preserves the locality of the features only in the data space. Due to this, the similarity between two local features belong to the similarity of their spatial neighborhood in the image. To overcome this flaw, we propose the integration of similarity based on Kullback-Leibler and wavelet decomposition in the domain of an image. This technique may surmount those limitations by taking into account each element of an image and its neighbors in similarity calculation. Classifications rates given by our approach show a clear improvement compared to those cited in this article.

Keywords

Sparse coding, features quantization, image representation, Laplace sparse coding, Kullback-Leibler , wavelet decomposition.

1. INTRODUCTION

Computer vision applications have experienced a great revolution with the integration of sparse coding techniques. Unfortunately, those techniques have not be able to model the locality and the similarity among the instances to be encoded owing to the overcomplete codebook and the independent coding process. Several approaches have been proposed to overcome this limitation. In [Gao10a], Gao proposed a method called Laplacian Sparse Coding which exploits the dependence among local features. Specifically, he suggested using histogram intersection based K-NN method to build a Laplacian matrix, which will characterize the similarity of local features. Furthermore, Laplacian matrix will be incorporated into the function of sparse coding to maintain the consistence in sparse representation of those features. In [Gao10b], Gao improved the technique of Kernel Sparse Representation. It is essentially the sparse coding technique in a high dimensional feature space mapped by implicit mapping function. In 2013, he proposed the Hypergraph Laplacian Sparse Coding techniques [Gao13]. In this case, he extracts the similarity between the instances within the same hyperedge simultaneously and also composes their sparse codes similar to each other.

In this paper, we propose an amelioration of the Laplacian sparse coding technique by changing the manner of similarity computation. In our case, the calculation of similarity in the image domain is based on the divergence of Kullback-Leibler and wavelet decomposition. This idea comes from its capacity to take into account neighbors' similarity.

This paper is composed of four sections. In section1, we introduce the Laplacian sparse coding technique. Section 2 describes the kernel sparse representation. We explain our approach in section3 and in the last one, we evaluate our approach.

2. Laplacian sparse coding

Sparse coding technique was proposed in order to reduce the problem of hard quantization. It solves the problem by proposing a sparse linear combination of basis vectors for each image feature. Sparse coding looks for a linear reconstruction of a signal $x,(x \in IR^d)$ using the bases in the codebook $U = (u_1, u_2, ..., u_k), (U \in IR^{d \times k})$.

The matrix of the sparse codes is $V = (v_1, v_2, ..., v_n)$ where $v_i \in IR^{K \times 1}$ and v_{ik} is the weight of the vector x_i in the basis vector u_k , the optimization problem of sparse coding can be reformulated as follows:

$$\min \|v\|_0 \; subject\ to\ x = UV \; or \min_{U,V} \|X - UV\|_F^2 + \lambda \sum_i \|v_i\|_1$$

$$\min_{U,V} \|X - UV\|_F^2 + \lambda \sum_i \|v_i\|_1 \; subject\ to \|u_j\| \le 1; \forall j = 1,...,K$$

λ is the tradeoff parameter used to balance the sparsity and the reconstruction error.

Because of the independent encoding feature resulting from an overcomplete or sufficient codebook. Asuming that $X = (x_1, x_2, ..., x_n)$ the vector of features, W is the matrix of similarity having W_{ij} the measuring of similarity of the pair (x_i, x_j). D the matrix of degree defined by $Dii = \sum_{j=1}^{n} W_{ij}$ is a diagonal matrix.

The Laplacian Sparse Coding, proposed in [Gao10a], takes into account the similarity between images both in features and image domains. The expression of Laplacian Sparse Coding is as follows:

$$\min_{v_1,...,v_n} \sum_i \|x_i - Uv_i\|_F^2 + \lambda \sum_i \|v_i\|_1 + \frac{\beta}{2} \sum_{ij} \|v_i - v_j\|^2 W_{ij} \quad (1)$$

This expression is defined by

$$\min_V \sum_i \|X - UV\|_F^2 + \lambda \sum_i \|v_i\|_1 + \beta tr(VLV^T) \quad (2)$$

Taking into account the Laplacian definition $L = D - W$ [Lux07].

Since the codebook U is not optimal , the expression can be rewritten as follows :

$$\min_{U,V} \sum_i \|X - UV\|_F^2 + \lambda \sum_i \|v_i\|_1 + \beta tr(VLV^T) \quad (3)$$

3. KERNEL SPARSE CODING

To ameliorate the technique of features representation using sparse coding, Gao proposed another approach called Kernel Sparse Representation. He noticed that kernel trick can pick up the nonlinear similarity of features. Kernel Sparse Representation is basically the sparse coding approach in a high dimensional feature space mapped by implicit mapping function [Gao10a] [Gao10b].

With the same consideration of sparse coding, we assume that there exists a feature mapping function $\Phi: IR^d \to IR^k, (d < k)$ with

$x \to \Phi(x)$,

$U = (u_1, u_2, ..., u_k) \to U_\Phi = (\Phi(u_1), \Phi(u_2), ..., \Phi(u_k))$.

According to this formulation, the expression of Kernel Sparse Coding is written as follow:

$$\min_{U,v} \|\Phi(x) - U_\Phi v\| + U_\Phi \|v\|_1 \quad (4)$$

Gao used Gaussian kernel due to its excellent performance in many works [Chen10] [Don04].

4. PROPOSED APPROACH

4.1. General context of multiresolution wavelet decomposition

Multiresolution wavelet decomposition analyses an image in time and frequency domains together. For lower frequency, it offers poor time resolution and better frequency resolution. Whereas, for higher frequency, it offers poor frequency resolution and better time resolution.

A multiresolution analysis is a family of nested sub spaces $L^2(IR)$ noted $(V_j)_{j \in Z}$ which have the following properties:

$$Vj = \left\{ \sum_{n \in Z} a_n^j \Phi_{j,n} : a_n^j \in IR \right\},$$

$$V_{j+1} \subset V_j, \bigcap_{j \in Z} V_j = \{0\}, \quad (5)$$

$$\bigcup_{j \in Z} \overline{V}_j = L^2(IR)$$

Hypothesis (5) means that $(V_j)_{j \in Z}$ is a space generated by the family $(\Phi_{j,n})_{n \in Z}$. Its definition depends on the chosen topology for the functional space. We can define it more strictly as the adhesion of the finite space of linear combinations of functions $\Phi_{j,n}$. Thus the approximation of a signal f on the space V_j is:

$$Aj = \sum_n a_n^j \Phi_{j,n} \quad (6)$$

Coefficients a_n^j are calculated by performing a scalar product signal with the family features $\Phi_{j,n}$:

$$a_n^j = < f, \Phi_{j,n} > \quad (7)$$

To write the difference between two consecutive spaces V_j and V_{j+1}, the space W_{j+1} is generated by a function $\psi_{j,n}$:

$$W_j = \left\{ \sum_{n \in Z} d_n^j \psi_{j,n} : d_n^j \in IR \right\} \quad (8)$$

The functions $\psi_{j,n}$ have values on the space W_j that is complementary to V_j in V_{j-1}. We have the same translational properties, expansion on $\psi_{j,n}$ than on $\Phi_{j,n}$. The set of functions $\psi_{j,n}$ is called space details. Thus, the detail of the signal f in the space W_j is calculated as follows:

$$D_j = \sum_n d_n^j \psi_{j,n} \quad (9)$$

And the coefficients of details d_n^j are calculated by the following formula:

$$d_n^j =< f, \psi_{j,n} > \quad (10)$$

The signal of which is assumed to be represented on a basis of V_j. Apply wavelet transform to the $k \in IN$ scale returns to representing the signal to a base adapted to the direct sum:

$$V_k \oplus W_k \oplus W_{k-1} \oplus ... \oplus W_{j+1} \quad (11)$$

The series of spaces V_j being fitted and following any function $f \in L^2(IR)$ of size n, can be decomposed into the basis of wavelets and scaling functions:

$$f = \sum_{k=1}^{n/2^j} a_k^j \Phi_{j,k} + \sum_{i=1}^{j} \sum_{k=1}^{n/2^j} d_k^i \psi_{i,k} \quad with \quad j \le m \quad (12)$$

If we perform an analysis to the last level, f becomes:

$$f = A_n + D_n + ... + D_2 + D_1 \quad (13)$$

4.2. Multiresolution Laplacian sparse coding

Gao in [Gao10a] [Gao10b] [Gao13] tried to preserve similarity by adding the Laplacian capacity to sparse coding technique. Furthermore, he ameliorated his technique by adding the hypergraph technique to the Laplacian sparse coding where the similarity among the instances is defined by a hypergraph. In this case, this technique captures the similarity among the instances within the same hyperedge simultaneously, and also makes their sparse codes similar to each other as shown in (4).

Despite these contributions, the modeling technique based on sparse coding remained unable to cover all similarities between features. This Laplacian sparse coding approach analyses images spatially and does not focus on the details of each object. We can therefore say that the analysis is done in a superficial way. For this, we propose the multiresolution Laplacian sparse coding to deepen these analyses.

Based on multiresolution Laplacian sparse coding, the modeling of images takes into account the modification of the neighbors of each object of an image. This idea came from the modeling capacity based on the divergence of Kullback-Lebleir and wavelet decomposition.

4.3. Wavelet and Kullback-Leibler divergence

Wavelet transform of an image I is the analysis of the image by a family of functions $\{\psi_{j,k}\}_{j,k}$. It consists of a dilated and translated ψ function called mother wavelet. Because of the localization properties in space and frequency of the mother wavelet, the wavelet coefficient $w(I)_{j,k} =< \psi_{j,k}, I >$ provides information about the content of the image I around point k and in a frequency band near the scale j. If the image is relatively smooth, then the wavelet transform concentrates most of the spatio-frequency information of the image into a few large amplitude coefficients [Pir08].

As a first approximation, these coefficients are uncorrelated which leads to processing by thresholding and denosing the wavelet coefficients which is very effective in image compression. But in reality, the wavelet coefficients scales are correlated at different scale. For example the presence of a discontinuity along a curve is translated into a point of this curve k_0 by large coefficients at all scales $\forall j, w(I)_{j,k_0}$.

Dependency models between different coefficients have been proposed to improve the description of spatial structures [Gor05] [Hub81]. In particular there is a dependency between a wavelet coefficient $w(I)_{j,k}$ and its closest neighbor's ladder $(w(I)_{j-1,k})$.

Banerjee et al. showed that coefficient vectors statistics wavelet shape (equation 14) is used to characterize the spatial structures of a very different kind [Ban05].

$$w(I)_{j,k} = (w(I)_{j,k}, w(I)_{j-1,k}, w(I)_{j,k\pm 1_x}, w(I)_{j,k\pm 1_y}) \quad (14)$$

To do this, it simply adjusts a Gaussian mixture model for each phenomenon to describe the joint probability of these vectors. In this case, it is unclear what types of structures will be present in the submissions, it cannot therefore set a model. However, it is hoped that the distribution of these vectors will be representative of spatial structures present in the image. Consequently, it is important to define a measurement taking into account the joint probability neighborhoods vectors wavelet $w(I)_{j,k}$.

Given the variability of spatial structures that can be encountered in the residual, the choice of a parameterization would be difficult to justify. We propose to introduce similarity metrics without valid parameterization of the distribution of neighborhoods: the metrics derived from the information theory such as residual entropy neighborhoods, mutual information or the Kullback-Leibler divergence between distribution neighborhoods of wavelet coefficients of the two images.

Suppose a neighborhood $w(I)_{j,k}$ containing d coefficients. Distribution of all neighborhoods of the image is denoted by $p_{w(I)}$ and checks $p_{w(I)} : IR \to IR \, et \int_{IR^d} p_{w(I)}(x)\, dx = 1$.

The differential entropy of Shannon $p_{w(I)}$ is defined by:

$$H(p_{w(I)}) = -\int_{IR^d} p_{w(I)}(x)\log p_{w(I)}(x)dx \quad (15)$$

It measures the amount of information contained in this distribution. The Kullback-Leibler is a measure of similarity between the distributions $p_w(I_1)$ and $p_w(I_2)$.

$$D_{KL}(p_w(I_1) \| p_w(I_2)) = \int_{IR^d} p_w(I_1)(x)\log\frac{p_w(I_1)(x)}{p_w(I_2)(x)}dx \quad (16)$$

Based on equations (15) and (16), the Kullback-Leibler distance is expressed as a difference of entropies:

$$D_{KL}(p_w(I_1) \| p_w(I_2)) = H_x(p_w(I_1), p_w(I_2)) - H(p_w(I_1)) \quad (17)$$

Knowing that the cross-entropy is defined as follows [Pir08]:

$$H_x(p_w(I_1), p_w(I_2)) = \int_{IR_d} p_w(I_1)(x)\log p_w(I_2)(x)dx \quad (18)$$

The use of these measurements on the distributions of the intensity of pixels of an image gives good results in the field of segmentation and image realignment [Ban05] [Fuk90] [Koz87]. A Kullback distance in wavelet space was also proposed for the indexing problem in [Col05] [Leo05]. Specifically, in these two articles, the authors parameterize the distribution of the wavelet coefficients for each scale j by a generalized Gaussian and sum the Kullback distances obtained at each scale for the similarity between the two images.

We propose to study similar measures to determine the similarity between two images, but with two major differences. First, the wavelet coefficients at different scales are not independent. Now summing the Kullback distances at each scale corresponds to the supposed independence. We therefore consider the accompanying entropy coefficients, in particular those of the previously described neighborhoods. On the other hand, we do not parameterize distributions game. We suggest measuring the similarity between images $I1$ and I2 as follows [Pir08]:

$$S(I1, I2) = \sum_j \alpha_j D_{KL}(p_{w_j}(I_1) \| p_{w_j}(I_2)) \quad (19)$$

- ✓ $p_{w_j}(I_1)$ is the non-parametric distribution of the coefficients of neighborhoods wavelet image I_1 to scale j [Pir08].
- ✓ $\alpha_j > 0$ is normalization weight according to attach redundancy wavelet system used [Pir08].

Based on the expression of the sparse coding and equation (2), we introduce the formula of multiresolution sparse coding:

$$\min_{v_1,\ldots,v_n} \sum_i \|x_i - Uv_i\|_F^2 + \lambda\sum_i \|v_i\|_1 + \frac{\beta}{2}\sum_{ij}\|v_i - v_j\|^2 < S_{ij} > \quad (20)$$

Based on the same equation (2), matrix W adopted by Gao in [Gao13] is filled by the coefficients of similarity of Kullback-Leibler S.

Using equation (19) in implementation, Sylvain Boltz in [Bol06] proposed an estimator of the Kullback-Leibler as follows:

$$D_{KL}(\tilde{T}, \tilde{R}) \overset{KNN}{=} \log\frac{N_{\tilde{R}}}{N_{\tilde{T}} - 1} + d\mu_{\tilde{T}}(\log\rho_k(\tilde{R})) - d\mu_{\tilde{T}}(\log\rho_k(\tilde{T})) \quad (21)$$

$\tilde{T}$ *and* $\tilde{R}$ are a set of data. $N_{\tilde{T}}$ *and* $N_{\tilde{R}}$ are the number of samples. $\rho_k(s)$ is a radius equal to the distance to the k^{th} nearest neighbor of s excluding s itself. This estimation is based on k nearest neighbors K-NN.

This estimator of the Kullback-Leibler distance can be computed relatively quickly whatever the size of samples. It is more robust to the choice of the number of k nearest neighbors.

Using the definition of Laplacian as in Gao in [Gao13]. We get the same equation (3).

5. EXPERIMANTS RESULTS

5.1. UIUC-Sport Dataset

This dataset consist of 8 classes [Li07]. Each class contains a set of images described in the following table:

Type	**Rowing**	**Badminton**	**Polo**	**Bocce**
Number	250	200	182	137

Snowboarding	**Croquet**	**Sailing**	**Climbing**
190	236	190	194

Table 1. Description of UIUC-Sport Dataset

5.2. Corel 10 Dataset

Corel 10 dataset is composed of 10 classes [Lu09]. Each class contains 100 images. The ten classes are skiing, beach, buildings, tigers, owls, elephants, flowers, horses, mountains, and food.

5.3. Results

To compare our results with those of Gao in [Gao13], we have chosen the same basis and the same number of selected images. The following tables resume all results.

Method	Classification rate
HIK+one Class SVM [Wu09]	83.54±1.13
ScSPM (1024) [Yang09]	82.74±1.46
ScSPM (2048) [Yang09]	82.94±1.60
LLC (1024) [Wang10]	83.09±1.30
LLC (2048) [Wang10]	82.50±1.27
LScSPM (1024) [Gao13]	85.18±0.46
LScSPM (2044) [Gao13]	85.27±1.14
Our approach	86.55

Table 2. Classification rate based on of UIUC-Sport Dataset

Table 2 evaluate classification rates obtained with different methods. Results of comparison have shown that our approach performs better results than the LScSPM [Gao13] approach in the context of classification applied to UIUC sport dataset.

Method	Classification rate
Spatial Mismatch Kernel [Lu09]	90.0
Spatial Markov Model [Lu09]	77.9
ScSPM [Yang09]	86.6±1.01
LLC [Wang10]	87.93±1.04
LScSPM [Gao13]	88.76±0.76
LScSPM+CM [Gao13]	91.86±0.89
Our approach	92.91

Table 3. Classification rate based on of Corel 10 Dataset

In table 3, we compared our technique to six other techniques in case of classification application. Results illustrated in the same table showed that multiresolution Laplacian sparse coding is the best technique of image representation for classification application.

6. CONCLUSION

In this paper, we suggested an improved method of image representation based on Laplacian sparse coding and the divergence of Kullback Leibler and wavelet decomposition. This measure of similarity is calculated between images which combine the concepts of information theory and wavelet transform. The principle of this approach is to sum the Kullback distances of each scale distributrion called neighborhood vectors of wavelet coefficients. The neighborhood coefficients, containing not only spatial locations but also relative scales, capture the spatial dependencies and inter-scale coefficients which can detect finer spatial structures. The Kullback distance on these vectors is estimated in a non-parametric manner despite their higher dimension, thanks to entropy estimators of nearest neighbors.

7. ACKNOWLEDGMENTS

The authors would like to acknowledge the financial support of this work by grants from General Direction of Scientific Research (DGRST), Tunisia, under the ARUB program.

8. REFERENCES

[Gao10a] Gao ,S. Wang, Z. Chia, L.T. and Tsang, I.W.H. Automatic Image Tagging via Category Label and Web Data, Proc. ACM Int'l Conf. Multimedia, 2010.

[Gao10b] Gao, S. Tsang, I.W. and Chia, L.T. Kernel Sparse Representation for Image Classification and Face Recognition, Proc. European Conf. Computer Vision, 2010.

[Gao13] Gao, S. Tsang, I.W. and Chia, L.T. Laplacian Sparse Coding, Hypergraph Laplacian Sparse Coding, and Applications, IEEE Transactions on Pattern Analysis and Machine Intelligence, VOL. 35, NO. 1, 2013

[Lux07] Luxburg, U.V. A Tutorial on Spectral Clustering, Statistics and Computing, vol. 14, no. 7, pp. 395-416, 2007.

[Chen10] Chen, X. Lin, Q. Kim, S. Pen, J. Carbonell, J.G. and Xing, E.P. An Efficient Proximal-Gradient Method for Single and Multi-Task Regression with Structured Sparsity, technical report, arXiv:1005.4717, 2010.

[Don04] Donoho, D. For Most Large Underdetermined Systems of Linear Equations, the Minimal l1-Norm Near-Solution Approximates the Sparsest Near-Solution, technical report, Stanford Univ., 2004.

[Pir08] Piro, P., Anthoine, S., Debreuve, E., Barlaud, M., Image retrieval via Kullback-Leibler divergence of patches of multiscale coefficients in the KNN framework, International Workshop on Content-Based Multimedia Indexing, PP:230-235, 2008.

[Gor05] Goria, M.N. Leonenko, N. N. Mergel, V. V. and Inverardi, P. L. N. A new class of random vector entropy estimators and its applications in testing statistical hypotheses. J. Nonparametr. Stat., 17(3):277–297, 2005.

[Hub81] Huber, P. Robust Statistics. John Wiley and Sons, 1981.

[Ahm76] Ahmad, I. and Lin, P. E. A nonparametric estimation of the entropy for absolutely

continuous distributions. IEEE Trans. Inform. Theory, 22(3):372– 375, 1976.

[Ban05] Banerjee, A. Merugu, S. Dhillon, I. and Ghosh, J. Clustering with bregman divergences. J. Mach. Learn. Res., 6:1705– 1749, 2005.

[Fuk90] Fukunaga, K. Introduction to statistical pattern recognition (2nd ed.). Academic Press Professional, Inc., 1990.

[Koz87] Kozachenko, L. and Leonenko, N. On statistical estimation of entropy of random vector. Problems Infor. Transmiss., 23(2):95–101, 1987.

[Col05] Collins, R. Zhou, X.and The, S. K. An open source tracking testbed and evaluation web site. In IEEE International Workshop on Performance Evaluation of Tracking and Surveillance (PETS), Breckenridge, CO, http://www.vividevaluation.ri.cmu.edu/, 2005

[Leo05] Leonenko, N. Pronzato, L. and Savani, V. A class of Renyi information estimators for multidimensional densities. PASCAL archive: http://eprints.pascalnetwork.org/archive/00001031/, 2005.

[Bol06] Boltz, S. Wolsztynski, E. Debreuve, E. Thierry, E. Barlaud, M. and Pronzato, L. A minimum-entropy procedure for robust motion estimation. In ICIP, Atlanta, GA, October 2006.

[Li07] Li, L.J. and Fei-Fei, L. What, Where and Who? Classifying Events by Scene and Object Recognition, Proc. IEEE Int'l Conf. Computer Vision, 2007.

[Lu09] Lu, Z and Ip, H.H. Image Categorization with Spatial Mismatch Kernels, Proc. IEEE Conf. Computer Vision and Pattern Recognition, 2009.

[Wu09] Wu, J. and Rehg, J.M. Beyond the Euclidean Distance: Creating Effective Visual Codebooks Using the Histogram Intersection Kernel, Proc. IEEE Int'l Conf. Computer Vision, 2009.

[Yang09] Yang, J. Yu, K. Gong, Y. and Huang, T. Linear Spatial Pyramid Matching Using Sparse Coding for Image Classification, Proc. IEEE Conf. Computer Vision and Pattern Recognition, 2009.

[Wang10] Wang, J. Yang, J. Yu, K. Lv, F. and Gong, Y. Locality-Constrained Linear Coding for Image Classification, Proc. IEEE Conf. Computer Vision and Pattern Recognition, 2010.

[Lu09] Lu, Z. and Ip, H.H. Image Categorization by Learning with Context and Consistency," Proc. IEEE Conf. Computer Vision and Pattern Recognition, 2009.

An EM based approach for motion segmentation of video sequence

Wei Zhao
Department of Data Science and Knowledge Engineering
Maastricht University
Maastricht, Netherlands
wei.zhao@maastrichtuniversity.nl

Nico Roos
Department of Data Science and Knowledge Engineering
Maastricht University
Maastricht, Netherlands
roos@maastrichtuniversity.nl

ABSTRACT

Motions are important features for robot vision as we live in a dynamic world. Detecting moving objects is crucial for mobile robots and computer vision systems. This paper investigates an architecture for the segmentation of moving objects from image sequences. Objects are represented as groups of SIFT feature points. Instead of tracking the feature points over a sequence of frames, the movements of feature points between two successive frames are used. The segmentation of motions of each pair of frames is based on the expectation-maximization algorithm. The segmentation algorithm is iteratively applied over all frames of the sequence and the results are combined using Bayesian update.

Keywords

Motion segmentation, EM algorithm, Bayesian update, SIFT feature, trajectory clustering

1 INTRODUCTION

Moving object detection is an important issue in the field of computer vision and one of the basic tasks of video processing. It differs from the class-specific object detection [Yan0, Fer03] and static object detection [Fel10, Pap98], which focus on building models of objects or background. Moving object detection is based on the assumption that foreground objects are usually accompanied by unique motion patterns [Hua07]. Techniques of moving object detection are widely used in different areas, such as video surveillance systems [Jos12], robot navigation [Jun04, Cal07], unmanned aerial vehicles [Rod12], and so on. In general, they consist of three main steps: motion detection, motion segmentation and object classification.

The motion detection can be achieved by tracking feature points [Wan13], or estimating the optical flow between frames to recover the motion of each image pixel [Cal07]. Motion segmentation aims at dividing the points (or pixels) into a set of groups according to their motion coherence [**?**, Vid04, Rod12, Jos12, Zha16]. The segmention results are groups of features points, or regions of images, which are processed by an object classification algorithm.

The approach proposed in this paper aims at segmenting moving objects in image sequences (videos). There are four steps in our approach: feature extraction, motion detection, motion segmentation, and combining segmentations of multiple frames, where the third and fourth steps are the key issues of the approach. Feature extraction and motion detection are realized by technique of scale-invariant feature transform [Low99]. The feature points are segmented into different groups based on their movements between pairs of image frames, using an adapted EM algorithm. The segmentations of multiple frame pairs are combined using Baysian update. The resulting groups of feature points are either moving objects in the scene or background regions. These groups of feature points can be processed by a classification algorithm. We evaluated our work in two ways: the accuracy of the segmentation and the computational efficiency.

A brief review of some related work will be given in the next section. The general architecture of the proposed model and the details of the segmentation algorithm of our approach are described in Section 4 . Experiments that we used to evaluate our approach are presented in Section 5. Section 6 concludes the paper.

2 RELATED WORK

Detecting and tracking moving objects is a challenging issue in the field of computer vision and a very important step in video processing. Numerous approaches of video-based object detection and tracking are proposed for different domains of use. Approaches for detecting similarities are widely used for moving object detection, which is related to techniques such as: optical flow, feature tracking, data clustering and segmentation [See13]. In this paper, we focus on the motion segmentation techniques.

Wang and Andelson used optical flow for motion estimation and k-means clustering for segmenting [Wan94]. Shi and Malik [Shi98] construct a weighted spatio-temporal graph on an image sequence and use normal cuts for motion segmentation.

Jung and Sukhatme [Jun04] proposed a moving objects detection system for mobile robots. They subtract the background by estimating the motion model of the camera. Pan and Ngo proposed to combine optical flow estimation with the EM algorithm [Pan05] for the purpose of image stabilization.

Vidal and Hartley proposed a motion segmentation algorithm for trajectory clustering by using generalized principal component analysis (GPCA) to cluster projected data [Vid04]. Jung and Sukhatme [Jun04] proposed a moving object detection system for mobile robots, where a probabilistic model accompanied with an adaptive particle filter and an EM algorithm is used for detecting the moving foreground objects. Elhamifar and Vidal use the sparse representation to cluster trajectories from multiple linear or affine subspaces [Elh09].

3 PRELIMINARIES

The approach proposed in this paper makes use of scale-invariant feature transform (SIFT), affine transformation, expectation-maximization (EM) and Bayesian update. In the section, a brief review of the techniques used in our paper is provided.

3.1 Scale-Invariant Feature Transform

SIFT is an algorithm to detect and describe local features in images, which was proposed by [Low99]. It is proved to be an efficient and robust way of detecting points of interests, which is useful in object detection and recognition. The SIFT feature are invariant to image scaling and rotation, and robust to large amounts of pixel noise [Low04]. Because of the scale-invariant properties and the high level feature expression, SIFT features are easy track in video sequences. Moreover, object recognition based on SIFT feature performs well [Low04].

3.2 Affine Transformation

The motions of objects in 3D space are projected to 2D images by camera in daily life videos. In a very short period, the changes of objects due to the 3D motions will be small and can be ignored. Thus the points belonging to one object can be assumed have the same 2D motions in frames. In that case, an affine transformation model is able to describe the movement of an object. If a point is detected at position x in one frame and at position x' in the next frame, then Equation 1 is assumed to hold for all points belonging to the same object.

$$\mathrm{x}' = A\mathrm{x} + b; \tag{1}$$

where $A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}$, $b = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}$.

3.3 Expectation-Maximization algorithm

The EM algorithm [Dem77] is an effective and popular technique for estimating parameters of a distribution from given data set.

Given the observed data $\mathbf{x}$ associated with a set of unobserved latent data or missing values $\mathbf{Z}$, and a vector of unknown parameters θ, the maximum likelihood estimate (MLE) of the unknown parameters is determined by maximizing the expected value of the likelihood function $L(\theta;\mathbf{x},\mathbf{Z}) = P(\mathbf{x},\mathbf{Z} \mid \theta)$.

Two steps are iteratively applied to find the MLE of the marginal likelihood until convergence,

E-step Given the parameters θ and the data $\mathbf{x}$ we can determine the probability distribution of the hidden variables $\mathbf{Z}$.

M-step Find a maximum likelihood estimate of the parameters.

$$\begin{aligned} &\theta = \mathrm{argmax}_{\theta'} L(\theta';\mathbf{x}) \\ &\text{where: } L(\theta;\mathbf{x}) = P(\mathbf{x} \mid \theta) = \sum_{\mathbf{Z}} P(\mathbf{x},\mathbf{Z} \mid \theta) \end{aligned} \tag{2}$$

In the application, we make use an adapted version to find hidden variables and parameters θ. Instead of the probability distribution $P(\mathbf{x},\mathbf{Z} \mid \theta)$ we determine:

$$\mathbf{z} = \operatorname*{argmax}_{\mathbf{Z}} P(\mathbf{x},\mathbf{Z} \mid \theta) \tag{3}$$

in the expectation step. In the maximization step we determine:

$$\theta = \operatorname*{argmax}_{\theta'} L(\theta';\mathbf{x},\mathbf{z}) \tag{4}$$

4 METHOD

We proposed a new approach for the segmentation of moving objects from video sequences. Fig.1 gives the architecture of our approach.

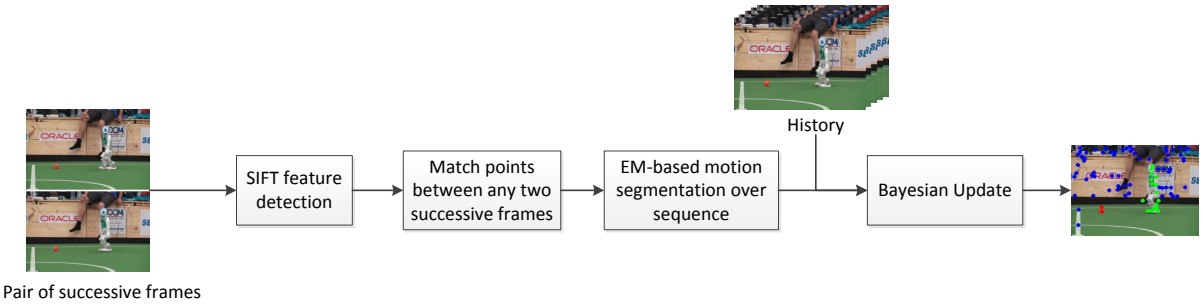


Figure 1: Architecture of our approach for video segmentation

We detect the SIFT feature points of each frame in the video sequence. The feature points of two successive frames are matched using the algorithm suggested by Lowe [Low04]. The movements of the matched feature points between two frames are subsequently obtained.

The movement of a point over multiple frames can be represented as a trajectory, which is a vector consisting of the positions of the point in multiple frames. Trajectories can be generated for "continuity" feature points, which means they appear in all frames of the sequence [Sun09, Wan13]. However, for many points, the "continuity" doesn't hold because of occlusion or 3D rotation of objects. Thus many feature points are excluded when requiring full trajectories over a sequence, which reduces the segmentation quality and increase the difficulty of recognition in the next step.

We investigate an segmentation algorithm making use feature points of both "continuity" and "discontinuity". An EM based segmentation algorithm is iteratively applied to segment feature points for each pair of successive frames. The segmentations are iteratively refined frame by frame using Bayesian update.

4.1 SIFT based motion detection

We detect the SIFT key points in each frame of the video sequence using the approach proposed by Lowe [Low04]. The movements of SIFT features can be identified by matching the corresponding features of two frames using the nearest-neighbours approach. The similarities of two features points are evaluated by computing the Euclidean distance between the feature vectors. A SIFT feature vector D_1 is matched to a SIFT feature D_2 only if the distance satisfy the following two conditions:

- The distance is smaller than some threshold.
- The distance is not greater than the distance of D_1 to all other descriptors.

RANSAC [Fis81] is used to refine the matching by filtering out the incorrect matches due to the imprecision of the SIFT model.

The movement vector of a matched point can be obtained by computing the displacement of the coordinates of matched the features, which denotes the position change of the same point in two different images. A motion flow field is determined by computing the movement vectors for all matched points. A motion field is generated between each pair of neighbouring frames.

4.2 Parametric Motion Model

An affine model of 6 parameters is used for representing the parametric motion model of an object. The affine model is estimated iteratively for movements between a pair of neighboring frames. Given the movement of any 3 points of the object, (A,b) can be computed. However, in our approach, the segments of moving points could contain outliers because the segmentation is not perfect. Moreover, the observed movements of points can contain noise. Given a set of pairs of feature points G, the parameters of affine model for one object can be estimated by solving the optimization problem:

$$\begin{gathered}(A,b) = \text{argmin}_{(A,b)} \sum_{(\mathbf{x},\mathbf{x}')\in G} ||\varepsilon||_{l_2} \\ \text{where } \varepsilon = \mathrm{x}' - A\mathrm{x} - b\end{gathered} \tag{5}$$

In some situations, the number of points belonging to an object is less than 3. For example, for a small rolling ball, SIFT can only detect 1 or 2 feature points on the ball. In this case, we assume the affine transformation degenerates to translation for one point, and a combination of translation and scaling for 2 points. The matrix A is reformulated as Equation 6.

$$A = \begin{cases} \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \text{ for group of 1 point} \\ \begin{pmatrix} a_{11} & 0 \\ 0 & a_{22} \end{pmatrix}, \text{ for group of 2 points} \\ \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}, \text{ for group of 3 or more points} \end{cases} \tag{6}$$

4.3 EM-based Motion Segmentation

Given the points and their movements between two frames, an EM-based segmentation algorithm is used to segment the motion field into groups of points each representing an object. Algorithm 1 gives the main steps of the EM based segmentation algorithm.

Algorithm 1 EM-based motion segmentation algorithm

if start frame **then**
  Initialize the points in one group
else
  Initialize the segmentation by reliabilities
end if
repeat
  repeat
    Using EM algorithm to estimate the best parameters of affine motion model, and the assignment of points
  until convergence
  if the group with the largest errors given the affine parameters exceeds the threshold **then**
    Split the group with the largest errors;
    Increase the number of objects by 1;
  end if
until no group can be find to split, or a maximum number of iterarions reached

In this algorithm, there are 3 components to be noticed:

Estimating affine parameters
Given a partition of points, the affine parameters of each group can be estimated by Equation 5 as discussed in Section 4.2.

Re-partitioning of points
Re-partitioning of points by reassigning the points to the groups, when the affine models are known. Suppose there are K groups, the division of points is regarded as an optimization problem:

$$\min \sum_{k\in[1,\ldots,K]} \mathscr{E}_k \tag{7}$$

where $\mathscr{E}_k = \sum_{(x,x')\in G_k} ||\varepsilon||_{l_2}$, and $\varepsilon = \mathrm{x}' - A\mathrm{x} - b$

Splitting
There are two aspects to be considered:

1. How to determine the group to be split?
   Given a partition of points, each group has an average error $\overline{\mathscr{E}_k} = \frac{1}{N_k}\mathscr{E}_k$ with respect to its motion model $(A,b)_k$. We choose the group with the largest average error to split.
2. How to split the selected group?
   We split the group with largest $\mathscr{E}_k$ using a bisecting K-means algorithm [**?**]. Once the group is split, a new partition of points and the corresponding motion models are computed. If the largest error of the new partition decreases, the current partition is updated by using the new partition and models. Otherwise, it means the optimal partition is found and no groups can be split, i.e. the iteration comes to an end.

4.4 Segmentation of trajectories

The EM-based segmentation algorithm in Section 4.3 deals with the temporal movements between two frames. It is extended to a video sequences using Bayesian update.

Given a image sequence of $T+1$ frames $f_0, f_1, \ldots f_T$, a segmentation is determined for each pair of successive frames (f_{i-1}, f_i). For each pair of frames, we estimated the probability $p(e|i,k)$ of the evidence e given the assignment of feature point i to a group k. Here the evidence is the error of the motion vector of a feature point with respect to the affine transformation of each group. We assume that the probability $p(e|i,k)$ is a decreasing function of the relative error of point i with respect to group k given K different groups. Equation 8 formalizes the computation of $p(e|i,k)$.

$$p(e|i,k) = 1 - \frac{\varepsilon_{i,k} + \frac{\delta}{K}}{\sum_{j=1}^{K} \varepsilon_{i,j} + \delta} \tag{8}$$

where $\delta = 0.1$, which is used for preventing dividing by zero.

Assuming that the evidence $E_t = (e_1, \ldots e_t)$ over t $(0 < t < T)$ pairs of frames in the sequence is independent, we may use Bayesian update to determine the probability that point i belongs to group k given all evidence E_t:

$$P(i,k|E_t) = \frac{P(E_t|i,k)}{P(E_t)} P(i,k) \tag{9}$$

where $P(i,k) = \frac{1}{K}$ and

$$\begin{aligned} P(E_t|i,k) &= P(e_1, \ldots e_t|i,k) \\ &= P(E_{t-1}|i,k) \cdot p(e_t|i,k) \end{aligned} \tag{10}$$

$$P(E_t) = \sum_{k=1}^{K} P(E_t|i,k) \tag{11}$$

5 EXPERIMENTS AND RESULTS

In this section, we will compare the segmentation results using our approach with some control approaches. Since our approach aims at dealing with long term motions, trajectory clustering algorithms of motion segmentation such as SSC [Elh09], GPCA [Vid04] and LSA [Yan0] are used for comparison. The segmentation is evaluated on video sequences from

Figure 2: Images from videos used in experiment

three data bases: the robocup 2014 video [1], CNnet 2014 [Wan14] and the Hopkins155 motion database[2]. There are videos of some indoor objects, moving pedestrian, moving cars, and robot soccer. Fig.2 shows some instance of the videos. Videos from Hopkins155 have a frame rate of 15 fps, while frame rate of videos from the other two data base are about 24 to 30 fps. Sequences of 30 frames are used in the experiments.

We will evaluate the approaches in the following ways:

- Evaluate the performance of segmentation algorithm on the data of motion trajectories.
- Evaluate the segmentation results w.r.t. all detected features.
- Evaluate the computing times.

Comparing with the other methods, our approach has the 3 unique characteristics:

- Our method includes the function of detecting feature points and their motions, while the comparison approaches are pure clustering algorithms for detected motion trajectories.
- Our method can deal with missing points in some frames, which doesn't hold for most of the comparison approaches because they require that the motion trajectories (the input data) are of the same length.
- Our method can determine the number of groups, while the other methods need the number of groups as an input.

Due to the differences of our method and the comparison methods, we designed two experiments to evaluate them. First, we will compare the performance of our motion segmentation algorithm on the full trajectories of the feature points provided by the data set Hopkins155, which will be discussed in Section 5.1. In this experiment, we don't detect feature points and their motions.

In the second experiment, we will evaluate our method using the original videos. The feature points and their motions are detected first. The segmentation quality over all detected SIFT features are evaluated, which will be discussed in Section 5.2.

All experiments are run on Matlab 2014a, with a computer of Intel Core i5 at 3.1GHz and 4GB of RAM.

5.1 Motion segmentation over trajectories

The experiment in this section runs on the Hopkins155 dataset. The codes for compared methods are from the site of hopkins155.

The Hopkins155 dataset contains 155 videos of 29 or 30 frames, each containing 2 or 3 moving objects. Each object is represented by a group of feature points. There are 266 to 398 feature points provided for each video, as well as the ground truth segmentation of the feature points. In these videos, the background is regarded as one object. Points from the background indicate the movement of the camera. The trajectory data $X \in \mathbb{R}^{2F\times N}$ is provided for each video, where F is the number of frames, N is the number of feature points. Each row of X is a trajectory of one feature point.

The videos are divided into 3 categories, the category named "checkerboard" contains several objects covered with a uniform checker board surface, which make 3D rotations and translations. The "traffic" sequences contain moving vehicles in outdoor traffic scenes. The remaining sequences named "articulated" contain motions constrained by joints, head and face motions, people walking, etc. Over half of the videos are taken using a moving camera.

Our segmentation method, named adapted EM segmentation using Bayesian update for motion sequences (AEM-b), is applied to the trajectories for segmenting the given feature points. The results are measured by the percentage of points that are clustered correctly, compared with the ground-truth clustering provided by the Hopkins155 dataset.

Table 1 shows the accuracy of segmentation results for sequences of different categories and number of motions. Each motion indicates an moving object (the background is also regarded as an object moving with the camera). The result of RANSAC for the same sequence can vary in each operation because of the statistical nature of RANSAC. We take the average results by running the algorithm 1,000 times for each sequence, and the threshold is set to 0.005.

In additional, we analysis the segmentation results of the category of 'checkerboard' videos, where the movements of camera varies in different situations. Our

[1] $https://www.youtube.com/watch?v=dhooVgC_0eY$
[2] $http://www.vision.jhu.edu/data/hopkins155/$

method performs 7% to 15% worse than SSC in this category. Table 2 shows the segmentation accuracy of different kind of videos according to the movements of camera. It is obviously that our method can achieve 99.8% in segmentation for the videos with a static camera. Our method is not good at deal with the videos taken by a rotating camera. More specifically, when we look into the details of the segmentation results of this category, our method performs very bad (under 70%) for the videos where the displacements of camera (both rotating and translating) is too large compare with the displacement of object itself.

Table 3 shows the accuracy of identifying the correct number of objects for our method.

The SSC outperforms all methods in general, while our method ranks 2nd out of 5 methods on average. We can draw the following conclusion from the results:

- AEM-b performs well for the traffic videos, where the major motions are 2d translations.
- AEM-b is able to find the number of objects automatically, with a high accuracy of 96.2%.
- AEM-b is not good at dealing with the 'checkerboard' videos, especially when the camera is rotating.
- AEM-b doesn't consider the relative position of feature points. Points apart from each other but with similar movements could be mis-clustered.

5.2 Motion Segmentation over detected points

In this section, we will apply the SIFT motion detection discussed in Section 4 directly to the original videos from CNet and robocup 2014. The SIFT features and their movements are generated frame by frame. For our method, we will apply the process of Figure 1, which will make use of the feature points existed in any two successive frames. Because the comparison approaches can only deal with trajectories of the same length for different lengths, we will detect the SIFT feature points existed in all frames, which will be result in a matrix of trajectories having the format of the data from Hopkins155 dataset.

The number of feature points in the trajectories matrix will decrease as the length of sequence increases. For each video, we test the methods using sequences of different lengths, which varies from 2 frames to 30 frames. Figure 3 shows the average number of feature points for different sequence lengths, with respect to different lengths of sequences. The blue line indicates all detected feature points, the red line is the number of feature points utilized by our method, and the green

LSA	RANSAC	GPCA	SSC	AEM-b
Checkerboard:78 sequences				
93.91	92.01	79.11	98.4	91.5
Traffic:31 sequences				
98.6	92.14	73.2	99.4	99.0
Articulated: 11 sequences				
96.9	90.45	72.5	98.9	92.0
All: 120 sequences				
95.4	91.9	77.0	98.8	93.5

(a) sequences with 2 motions

LSA	RANSAC	GPCA	SSC	AEM-b
Checkerboard:26 sequences				
68.1	72.23	80.4	97.4	83.9
Traffic:7 sequences				
80.2	88.28	53.1	99.2	98.9
Articulated: 2 sequences				
83.2	76.98	78.9	98.9	84.4
All: 35 sequences				
71.3	75.7	74.9	97.9	87.0

(b) sequences with 3 motions

LSA	RANSAC	GPCA	SSC	AEM-b
All:155 sequences				
90.0	88.2	76.5	98.5	92.1

(c) all sequences

Table 1: Accuracy (%) of motion segmentation for different settings

LSA	RANSAC	GPCA	SSC	AEM-b
Static camera: 20 sequences				
92.2	92.2	89.4	99.6	99.8
Translating camera: 20 sequences				
79.9	81.8	76.9	99.1	96.5
Rotating camera: 24 sequences				
71.0	76.4	62.7	98.0	80.1
Rotating and translating camera: 40 sequences				
94.2	93.4	83.2	97.5	90.4

Table 2: Segmentation of checkerboard videos according to the movement of camera

Sequences of	Checker-board	Traffic	Articulated
2 motions	92.8	96.6	81.2
3 motions	86.7	98.4	83.6
all		89.9	

Table 3: Accuracy (%) of estimating the number of objects

line shows the number of points utilized in the trajectories. It is clearly that our method can make use of more points in each pair of frames. The number of utilized feature points remains stable with growing length of sequences in our method, while it decreases sharply for trajectories.

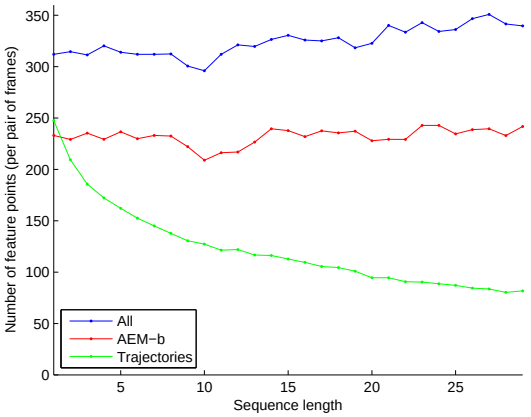


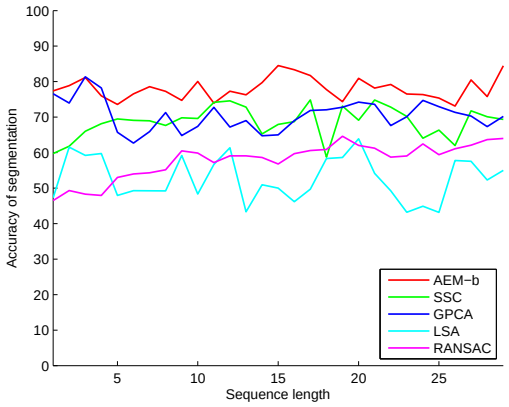
Figure 3: Number of feature points used in different methods

Figure 4a shows the segmentation accuracy of all methods with respect to all trajectories of the specified sequence lengths. Figure 4b shows the segmentation accuracy with respect to all feature points. Because the comparison methods are all using the trajectories as inputs, their segmentation accuracies w.r.t. all feature points decrease when sequences getting longer. From the Figure 4a and 4b, we can draw the following conclusions:

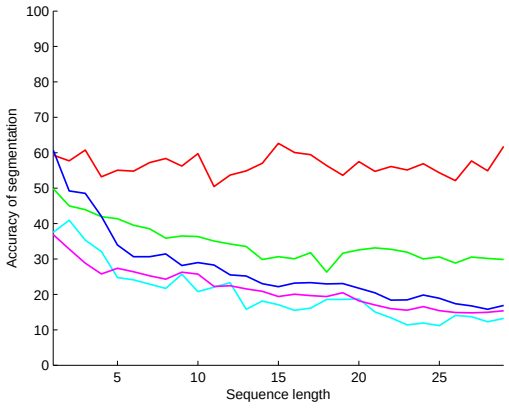
- For the original videos, our method provides a solution including the feature detection, motion estimation and segmentation, which can make use of more feature points. The other methods require a separate step of building trajectories, which will lead to a loss of feature points.

- Our method can achieve higher accuracy of segmentation in the videos from CNet and robocup 2014, where the movement of feature points are not as accurate as them from the Hopkins155 dataset. For the latter one, the movements of features points are detected by a special tracker.

- Our method can always make use of the most feature points even the length of sequence increases. Thus the accuracy of segmentation compared to all detected points are relatively stable compare to the other methods. More feature points will profit the next step of object recognition.



(a) Accuracy of segmentation



(b) Accuracy of segmentation w.r.t. all feature points

Figure 4: Accuracy curves w.r.t. lengths of sequences, compare to (a) the utilized points (b) all feature points.

5.3 Computing time

Table 4 and 5 give the average computing time for sequences with a length of 30 frames for different dataset. Although RANSAC and GPCA have the lowest computation times, their segmentation accuracy is also lower. Moreover, the performance of RANSAC is not stable as mentioned in Section 5.1. Our method has an average computation time of 0.3s, which is smaller than LSA and SSC.

	LSA	RANSAC	GPCA	SSC	AEM-b
Number of points			330		
Hopkins155	4.32s	0.09s	0.14s	3.8s	0.31s

Table 4: Computing time (seconds per 30 frames) of segmentation stage of Hopkins155 dataset

For experiment two, we only consider the computation time of the segmentation stage, which means the computing time of feature detecting and motion estimation is not taken into consideration.

	LSA	RANSAC	GPCA	SSC	AEM-b
Number of points	78	78	78	78	235
CNnet	0.94s	0.01s	0.04s	0.68s	0.19s
RobotCup	0.91s	0.01s	0.04s	0.66s	0.20s

Table 5: Computing time (seconds per 30 frames) of segmentation stage of CNet and robocup2014 videos

	LSA	RANSAC	GPCA	SSC	AEM-b
Hopkins155	13.0	0.27	0.42	11.4	0.9
CNnet	12.1	0.19	0.51	8.72	0.8
RobotCup	11.7	0.19	0.51	8.46	0.9
all	12.2	0.2	0.5	9.5	0.9

Table 6: Computing time (ms per point per 30 frames) of segmentation stage

In Figure 3 we can see that the average number of points utilized in trajectory clustering for a 30 frame sequences is about 80, while it is about 240 in our method. That means our methods will process three times more points compared to the other methods in this experiment. Nevertheless, our method is faster than SSC and LSA. Table 6 shows the averge computation time per feature point. Taking the difference in the number of feature points in to account, our method is ten times faster than SSC, fourteen times faster than LSA, three times slower that RANSAC, and two times slower than GPCA.

6 CONCLUSION

We proposed an approach for segmenting video frames into groups of feature points based on their motions. In the proposed method, SIFT feature points and their movements are detected using Lowe's algorithm [Low04], an adapted EM algorithm is applied with a recursive division strategy for segmenting the feature points according to their motions. The segmentation is iteratively applied for each pair of frames in the sequence, and combined with Bayesian update to generate segmentation results over all frames. The characteristics of our method are as follows

- Because our method processes pairs of frames iteratively, it can deal with arbitrary length of video sequence.
- The EM algorithm with a division strategy can determine the number of moving objects in the frames.
- Bayesian update combines the results of a sequence of frames.
- Our method can handle the problem of missing points in any frames, because it does not track feature points over sequence of frames. We only consider the feature points in neighbouring frames in each step of the segmentation.

Results shows that our method performs well in trajectory segmentation, and has a average accuracy of 92.1% in general. It is especially successful for videos of translation. However, it performs not well the displacement of objects is small compared to the displacement caused by the moving camera. Our approach does not require that all trajectories of feature points have the same length, which means that it can deal with the data with missing points. This property makes our approach more flexible than other approaches. Experiments also show that the computational cost of our method is reasonable. On the one hand, it performs better than the methods which are faster. On the other hand, it is ten times faster than the methods perform better (actually only the SSC) in the segmentation stage giving the trajectories of feature points (provided by Hopkins155 dataset). In general, our method proposes an efficient way to deal with motion segmentation of video sequences in a dynamic environment.

The first drawback of our method is that it can not deal very well when the movement of camera is significant compare with the movements of the objects. Secondly, our method does not consider the position relationships of points, so some points being far away from an object but having similar movements will be misclassified, which is not a big problem for SSC. Thirdly, the performance of our method drops too much when the number of moving objects increases, compare to the best one (SSC).

In the future, we will do more experiments to evaluate the robustness of our methods in varying conditions. The motion model should be made more robust for camera movements. Exploring whether different types of feature points influence the segmentation is also worth investigating. Last but no least, we will investigate its applicability in real time for mobile robots.

7 REFERENCES

[Bor97] Borshukov, G. D., Bozdagi, G., Altunbasak, Y., and Tekalp, A. M. . Motion segmentation by multi-stage affine classification. IEEE Trans. Image Processing, 6:1591-1594, 1997.

[Cal07] Caldeira, E. M. d. O., Schneebeli, H. J. A., and Sarcinelli-Filho, M. An optical flow based sensing system for reactive mobile robot navigation. Sba: Controle & Automação Sociedade Brasileira de Automatica, 18(3):265-277, 2007.

[Dem77] Dempster, A. P., Laird, N. M., and Rubin, D. B.. Maximum likelihood from incomplete data via the em algorithm. Journal of the Royal Statistical Society. Series B (Methodological), pages 1-38, 1977.

[Elh09] Elhamifar, E. and Vidal, R. (2009). Sparse subspace clustering. In Computer Vision and Pat-

tern Recognition, CVPR 2009. IEEE Conference on, pages 2790-2797, 2009.

[Fel10] Felzenszwalb, P. F., Girshick, R. B., McAllester, D., and Ramanan, D. Object detection with discriminatively trained part-based models. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 32(9):1627-1645, 2010.

[Fer03] Fergus, R., Perona, P., and Zisserman, A. Object class recognition by unsupervised scale-invariant learning. In Computer Vision and Pattern Recognition. Proceedings. 2003 IEEE Computer Society Conference on, volume 2, pages II-264, 2003.

[Fis81] Fischler, M. A. and Bolles, R. C. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Communications of the ACM, 24(6):381- 395, 1981.

[Hua07] Huang, S.-S., Fu, L.-C., and Hsiao, P.- Y. (2007). Region-level motion-based background modeling and subtraction using mrfs. Image Processing, IEEE Transactions on, 16(5):1446-1456.

[Jos12] Joshi, K. A. and Thakore, D. G. A survey on moving object detection and tracking in video surveillance system. IJSCE, ISSN, pages 2231-2307, 2012.

[Jun04] Jung, B. and Sukhatme, G. S. Detecting moving objects using a single camera on a mobile robot in an outdoor environment. In International Conference on Intelligent Autonomous Systems, pages 980-987, 2004.

[Low99] Lowe, D. G. (1999). Object recognition from local scale-invariant features. In Computer vision. The proceedings of the seventh IEEE international conference on, volume 2, pages 1150-1157, 1999.

[Low04] Lowe, D. G. Distinctive image features from scale-invariant keypoints. International journal of computer vision, 60(2):91-110, 2004.

[Pan05] Pan, Z. and Ngo, C. W. Selective object stabilization for home video consumers. IEEE Trans. Consumer Electronics, 51(4):1074-1084, 2005.

[Pap98] Papageorgiou, C. P., Oren, M., and Poggio, T. A general framework for object detection. In Computer vision, 1998. sixth international conference on, pages 555-562, 1998.

[Rod12] Rodríguez-Canosa, G. R., Thomas, S., del Cerro, J., Barrientos, A., and MacDonald, B. A real-time method to detect and track moving objects (datmo) from unmanned aerial vehicles (uavs) using a single camera. Remote Sensing, 4(4):1090-1111, 2012.

[See13] Seerha, G. K. and Rajneet, K. Review on recent image segmentation techniques. International Journal on Computer Science and Engineering (IJCSE), 5:109-112, 2013. [Selim and Ismail, 1984] Selim, S. Z. and Ismail, M. A. (1984). K-means-type algorithms: a generalized convergence theorem and characterization of local optimality. Pattern Analysis and Machine Intelligence, IEEE Transactions on, (1):81-87.

[Shi98] Shi, J. and Malik, J. (1998). Motion segmentation and tracking using normalized cuts. In Computer Vision. Sixth International Conference on, pages 1154-1160, 1998.

[Sun09] Sun, J., Wu, X., Yan, S., Cheong, L.- F., Chua, T.-S., and Li, J. Hierarchical spatiotemporal context modeling for action recognition. In Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on, pages 2004-2011, 2009.

[Vid04] Vidal, R. and Hartley, R. Motion segmentation with missing data using powerfactorization and gpca. In Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on, volume 2, pages II-310, 2004.

[Wan13] Wang, H., Kläser, A., Schmid, C., and Liu, C.-L. Dense trajectories and motion boundary descriptors for action recognition. International journal of computer vision, 103(1):60-79, 2013.

[Wan94] Wang, J. Y. and Adelson, E. H. (1994). Representing moving images with layers. Image Processing, IEEE Transactions on, 3(5):625-638, 1994.

[Wan14] Wang, Y., Jodoin, P.-M., Porikli, F., Konrad, J., Benezeth, Y., and Ishwar, P. Cdnet 2014: An expanded change detection benchmark dataset. In Computer Vision and Pattern Recognition Workshops (CVPRW), 2014 IEEE Conference on, pages 393-400, 2014.

[Yan0] Yan, J. and Pollefeys, M. A general framework for motion segmentation: Independent, articulated, rigid, non-rigid, degenerate and nondegenerate. In Computer Vision-ECCV 2006, Springer, pages 94- 106, 2006.

[Zha16] Zhao, W. and Roos, N. Motion based segmentation for robot vision using adapted em algorithm. In Proceedings of the 11th International Conference on Computer Vision Theory and Applications (VISIGRAPP 2016), pages 649-656, 2016

ISSN 2464-4617 (print) WSCG 2016 - 24th Conference on Computer Graphics, Visualization and Computer Vision 2016
ISSN 2464-4625 (CD-ROM)

Procedural Texture Synthesis by Locally Controlled Spot Noise

Nicolas Pavie[1], Guillaume Gilet[1], Jean-Michel Dischler[2], Djamchid Ghazanfarpour[1]
[1] XLIM - UMR CNRS 7252, University of Limoges, France
[2] ICUBE - UMR CNRS-UDS 7357, University of Strasbourg, France

ABSTRACT

Procedural noises based on power spectrum definition and random phases have been widely used for procedural texturing, but using a noise process with random phases limits the types of possible patterns to Gaussian patterns (i.e. irregular textures with no structural features). Local Random Phase (LRP) Noise has introduced control over structural features in a noise model by fixing the frequencies and phase information of desired features, but this approach requires storing these frequencies. Space distortion and randomization must also be used to avoid repetitions and periodicity. In this paper, we present a noise model based on non-uniform random distributions of multiple Gaussian functions for synthesizing semi-structured textures. We extend the LRP noise model by using a spot noise based on a controlled distribution of kernels (spots), as an alternative formulation to local noises aligned on a regular grid. Spots are created as a combination of Gaussian functions to match either a specific power spectrum or a user-defined texture element. Our noise model improves the control over local structural features while keeping the benefits of LRP noise.

Keywords

Procedural texturing, Image synthesis, procedural noise

1 INTRODUCTION

Random signals have been widely used for procedural texturing since the marble pattern of Perlin [Per85]. Noise-based procedural textures inherit many properties of procedural noises, the most compelling ones being :

- No repetition is visible;
- The pattern produced is continuous over its evaluation space;
- It can be computed during rendering on a per-pixel basis;
- One texture model can produce various patterns by tuning parameters

These advantages have led to a growing study of noise applications in procedural texturing. A large variety of patterns can be produced by a noise-based process by defining a given power spectrum (Gabor noise [LLDD09], Multiple Kernel noise [GDG12b]) but shaping a pattern by directly tuning the spectrum of a noise remains a difficult task, because the correlation between a target pattern and the corresponding power spectrum cannot be straightforwardly deduced. A more artist-friendly approach consists in computing the noise parameters from a pattern sample [GLLD12].

Most of the recent "noise by example" methods consider a given image as an input and aim at generating a noise reproducing its power spectral density (PSD), computed by spectral analysis. Visual variety in the results is introduced by keeping the phase information random. However, it is well known that structure can be found in the phase information of the spectral analysis [OL81].

The recently introduced LRP Noise [GSV*14] tackles the problem of structural features preservation by fixing both phases and magnitudes in some areas of the frequency spectrum approximating the structural component of the input example. This approach has two drawbacks. Firstly, it requires to store the phase information of the relevant parts of the spectrum. Secondly, as only a limited number of fixed frequencies are used, periodicity of the structural components must be broken by using turbulence [Per85] and random shifts.

In this paper, we present an alternative formulation of the LRP noise based on a locally defined and controllable spot noise representation. We focus on structural features that can be defined by a repetitive structured kernel function (i.e. fabric textures with specific stitches aspects and random small variations). Such kernels can be created by an arrangement of the base components of the features (i.e. the threads within a stitch). This formulation retains the advantages of the

LRP noise : Both stochastic details and structural features can be generated in real-time on a per pixel basis. Simple structures can be reproduced through an automatic process. The benefit provided by our representation is two-fold :

- The sum of cosines modeling the structural component of the LRP Noise formulation is replaced by a sum of few Gaussian kernels. This compact representation reduces computation cost for local structures and yields a noise process with similar performance but with enhanced control over the final visual appearance.
- The distribution of the structural features of the input can be edited and is part of the definition of the model. This extends the range of possible patterns that can be produced, from very regular to completely stochastic ones, but still featuring structural components. Repetitions and periodicity are avoided for semi-regular patterns since the distribution of local noises is still based on a random process.

The control over both distribution and kernel aspect in the noise model allows for interactive edition of patterns.

A comparison of local noise formulations is presented in fig.1.We present the possibilities offered by our new formulation through several examples of patterns as shown in fig.3.

2 RELATED WORKS

2.1 Procedural patterns synthesis

To create procedural patterns, several approaches can be used depending on the desired degree of "randomness". For structured and semi-structured procedural patterns, patch-based approach can help artists expanding a pattern sample with characteristic structural features. In such approach, a procedural pattern is evaluated by tiling the surface with patches (small textures) [CSHD03, EF01, VSLD13]. Patches are randomly arranged to break repetitions, but results may lack of details variety : the same tiles / patches (i.e. rigorously identical contents) are repeated over and over again even for irregular textures .

Semi-structured pattern can also be synthesized as a distribution of objects in texture space [GD10]. To create a procedural pattern with this approach, a procedural distribution function is required to create an infinite set of random position. Point jittering is often used as distribution function for its simplicity and evaluation speed [Gla04]. But it does not take in account spatial dependencies (distance threshold between objects) so distributed objects may overlap. *Direct Stochastic tiling* [LD05] can produce some distance dependencies, to create for example an infinite set of Poisson-disks. But it still requires some tiles to be precomputed and stored. For their *assemblage* creation, [GDG12a] proposed an improvement of point jittering to take in account some spatial dependencies : the squared lattice is replaced by polygonal cells that forms a rectilinear tesselation of the plane. Similarly to jittering, each cell contains a different instance of an object with a random position computed on-the-fly. Fully procedural semi-structured pattern can be produced using both procedural objects definition and procedural distribution function, but very few techniques propose to extract such objects directly from an input sample. Irregular and near-regular patterns can also be generated with Markov Random Fields [CJ83]. [VGR16] specifically consider Markov-Gibbs Random Fields to create stochastic, irregular and near-regular textures. This approach can reproduce patterns with complex structural details from an example with great accuracy. But the texture generation processes associated with such models are highly iterative and focus on statistical reproduction over generation speed. It makes them unpractical to use in a rendering pipeline for high resolution textures generation on-the-fly.

To create procedural pattern with greater randomness, procedural noise functions are often preferred over spatial description methods (more details in section 2.2). But as modeling a power spectrum is no trivial exercise, several noises "by example" use a self-configuration process to approximate a specified power spectrum, within the noise spectral capabilities. [LVLD10] describe a process to reproduce isotropic patterns by decomposing a Power Spectral Density (PSD) into several frequency bands to compute the weights of a multi-resolution wavelet noise. [GDG12b] extract several kernel configurations from an arbitrary PSD by decomposing a spectral domain into sub-regions of specific magnitude range. As an extention of the Gabor noise, [GLLD12] also describe a method to reproduce an arbitrary PSD in several band-limited Gaussian spectrums. Each spectrum corresponds to a band limited Gabor noise. These noises are nonetheless limited to Gaussian patterns : as they are completely characterized by their power spectrum, only micro-structural features are produced. Local Random Phase noise [GSV*14] is of particular interest as it introduces structure preservation in its noise formulation while allowing the "by example" approach.

2.2 Procedural noises

Procedural noises have been widely used as a modeling tool for texture synthesis after the Perlin noise first appeared in [Per85]. A procedural noise implies no discrete data samples, a very low storage requirement (i.e. a simple evaluation function), no periodicity nor repetitions. Two families of procedural noises are gener-

ally considered (see survey [LLC*10]) : lattice gradient noises and sparse convolution noises. Lattice gradient noises are based on the interpolation of randomly oriented gradient ([EMP*02]) dispatched on a regular grid. Sparse convolution noises are based on the convolution of a spatial filter function (kernels) with a random distribution of impulses (points).

Random distribution processes result in a white noise in the frequency domain, so the control of sparse convolution noises can be achieved by spectral definition of the kernel function. A sparse convolution noise can be constructed around a specific evaluation functions such as Gabor-[LLDD09], Gaussian-[Lew89], or Sync-[GDG12b] kernels. The latter use multiple configurations of the kernel to optimize spectral coverage. A more spatial-oriented formulation of a sparse convolution noise was proposed by [vW91, dLvL97] with the Spot Noise. It is based on an arbitrary spatial kernels. Some micro-structural features can be produced by using structured kernel. But the quantity of the structural features produced remains limited by the random distribution process.

Local Random Phase noise [GSV*14] states that structural features are contained in both the magnitude and the phase spectrum of specific frequencies. To produce structures within a noisy pattern, LRP noise model propose to fix their corresponding frequencies. While it achieves to produce structures accurately, this noise model suffer from several drawbacks : 1) frequencies of structures selected for reproduction need to be stored; 2) Local cosine-based noises need a great number of cosines to cover the spectrum.

We extend the LRP noise formulation to produce a more compact representation of local structures by relying on a spot noise formulation. Our local spot noises use a sum of quadratic Gaussian functions to create structured or unstructured kernels, so a wide range of possible spot aspects can be produced. Locally defined structural features are further enhanced by the introduction of a constrained random distribution.

3 NOISE MODEL

We now present our alternative formulation of the LRP noise model based on spot noise. As a reminder, the original formulation of the LRP noise is the following

$$n(x) = \sum_{i=1}^{I} w\left(\frac{||x-x_i||}{\Delta}\right) \sum_{j=1}^{J} A_{i,j}\cos(2\pi f_{i,j}\cdot x + \rho_{i,j}) \tag{1}$$

It is a mix of $J \times I$ local cosine-based noises with random phases and windowed over a regular lattice (x_i is the position over the spatial lattice corresponding to a spectral stratum i) : randomness is obtained by the random phases while the spectrum is controlled by the frequencies sampling of each noise. We now propose an alternative formulation based on a fully procedural spot noise to limit the number of cosines. We also present a new procedural distribution function to control the locality of the noise produced. The extended range of patterns that can be produced using this distribution is presented in fig. 3.

3.1 Procedural Multiple-Gaussian spot noise

Sparse convolution noise [Lew89] is originally based on the random distribution of impulses convolved with an isotropic Gaussian kernel. Such kernels, created as the multiplication of a sample texture by a Gaussian envelope, only produce isotropic Gaussian patterns due to the fixed Gaussian envelope of the kernel used. To improve spectral control, Gabor kernel [LLDD09] can be used as it unifies spectral and spatial characteristics. But spatial control is reduced at the same time. It can produce only Gaussian textures, which is an excessively narrow subset of procedural patterns. Spot noise [dLvL97] can produce a wider range of patterns, including non-gaussian patterns containing structural features, by using an arbitrary spatial kernel instead.

We aim at spatial characteristics that cannot be produced by the sole power spectrum definition. [vW91] noted that structural characteristics present in the kernel itself, such as (an)isotropy or a structural feature, result in similar characteristics within the texture produced by the spot noise. In other words, when the kernel contains some structure, this structure is transferred to the texture. A formulation of spot noise is given by [dLvL97] as :

$$n_s(\mathbf{p}) = \sum_{j}^{J} w_j(\mathbf{p}_j)\, k_s\left((\mathbf{p}-\mathbf{p}_j)\,\mathbf{R}_s(\mathbf{p}_j)\,\mathbf{S}_s(\mathbf{p}_j)\right) \tag{2}$$

Where $\mathbf{p}_j$ is an impulse position, and k_s is the kernel function of the spot s. Orientation $\mathbf{R}_s$ and scale $\mathbf{S}_s$ are related to underlying data fields. Impulse positions $\mathbf{p}_j$ are uniformly distributed using a Poisson process and the weights w_j are equiprobably drawn in $[-1,1]$.

For our kernel formulation, we use a sum of ellipsoidal N-dimensional Gaussian functions with arbitrary scale and orientation. Gaussian functions (and kernels by extension) are commonly used in noise literature to create Gaussian noise patterns. It has also been used for modeling surfaces and volumes [JBL*06]. In computer vision, Gaussian kernels can be used as a reconstruction primitive after a Gabor-wavelet decomposition of an image [WM03]. For spot noise modeling, a wide range of kernels can be produced by combining a few simple Gaussian functions. Some examples of kernels are presented in figure 2. Our kernel k is defined for a dimension N as:

$$k(\mathbf{p}) = \sum_{V_i} g_{V_i} \qquad g_V(\mathbf{p}) = Ae^{-\frac{1}{2}\mathbf{p}^T\mathbf{V}^{-1}\mathbf{p}} \tag{3}$$

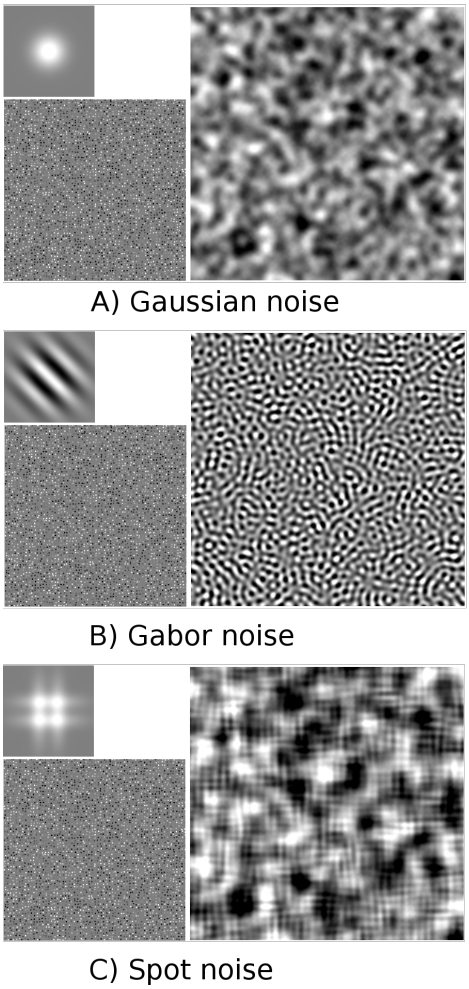


Figure 1: Examples of noises generated by sparse convolution, with their respective kernel and impulse distribution. Unlike Gaussian (a) and Gabor (b) noise, spot noise (c) can also produced semi-regular structural features using a single arbitrary spatial kernels.

Here A is the Gaussian magnitude, and $\mathbf{V}$ is a $(N+1)\times(N+1)$ matrix, such that $\mathbf{V}^{-1} = (\mathbf{M}\,\mathbf{R}\,\mathbf{S})^{-T}(\mathbf{M}\,\mathbf{R}\,\mathbf{S})^{-1}$ and $|\mathbf{V}|$ is the matrix determinant. $\mathbf{M}$,$\mathbf{R}$ and $\mathbf{S}$ respectively correspond to shift, rotation and scaling matrices. The isocountour of the kernel is given by $\mathbf{p}^T\mathbf{V}^{-1}\mathbf{p}$, which describes an implicit surface given that $\mathbf{V}$ is a semi-positive matrix ($\mathbf{p}$ is a point in dimension $N+1$ with last component set to 1). We show in figure 1.c an example where using a simple grid-kernel composed of four ellipsoidal Gaussian can effectively produce a texture with semi-regular structural features. With the sparse convolution process, the evaluation window is considered to be induced by the kernel formulation (i.e. each Gaussian function falls below a threshold before the maximum evaluation distance of a kernel). Considering this and equation 2, the new formulation of our complete noise model becomes :

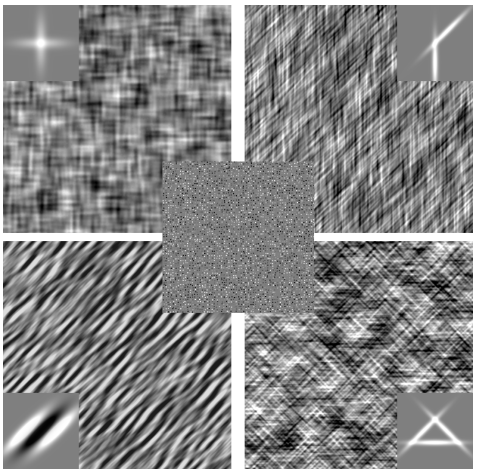


Figure 2: Spot noise with various kernel aspects, composed of several elliptic Gaussian function. Each oriented Gaussian within a kernel generates an oriented component within the results. The center tile shows their random distribution of impulses. Performances are around 85 fps for the 2-Gaussians spots (top row) and 65 fps for the 3-Gaussians spots (bottom row).

$$n(\mathbf{p}) = \sum_{i=0}^{I} w_i n_i(\mathbf{p}) \tag{4}$$

Where n_i is a local noise composed of random impulses of the kernel i. Here w_i is an energy normalization factor computed for each noise. Our noise model is a composition of several spot noises. Each spot noise can be used to model a specific set of features of a pattern.

Texture generated by this formulation produces local structural features (see the "grid-shaped" micro-pattern in figure 1(c)) while keeping the randomness introduced by the Poisson distribution of impulses. To further widen the range of possible patterns generated by this formulation, we propose to extend the previous spot noise by introducing a non-uniform random distribution of impulses.

Let $\mathbf{p} = (X,1)$, a point of coordinates X in a specified dimension, we propose a new formulation :

$$n_s(\mathbf{p}) = \sum_j \delta\left(\xi(\mathbf{p}_j) < d(\mathbf{p}_j)\right) |w_j(\mathbf{p}_j)| K_j(\mathbf{p}) \tag{5}$$

with $K_j(\mathbf{p}) = k_s\left((\mathbf{p}-\mathbf{p}_j)\,\mathbf{R}_s(\mathbf{p}_j)\,\mathbf{S}_s(\mathbf{p}_j)\right)$. d is a scalar field and represents a probability. δ denotes the Kronecker delta and $||$ the absolute value. ξ is a random variable selected independently of w_j. d allows us to control the density of impulses in given regions. Because we use an absolute value, high density regions imply noise values close to 1 whereas low density values result in values close to 0.

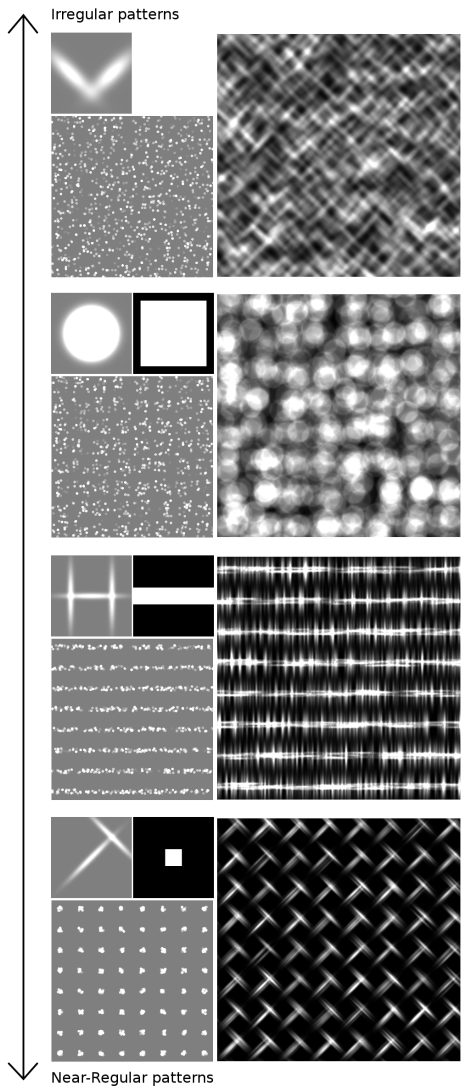


Figure 3: Examples of pattern produced with our distribution, from irregular (top row) to regular patterns. For each example, the top images of the the left column are the kernel (left) and the periodic density (right) profiles used, the bottom left image shows the resulting impulse distribution. The image of the right column shows the noise result.

The global appearance of the produced texture is directly correlated to the shape of d : the energy of the pattern is concentrated around higher density areas within the density field. It introduces a new level of control over the various appearance of the generated texture and can be used to introduce global structure at a large scale. To model structural regularity, we define d as a periodic density field tiling the evaluation space (i.e. fig. 3,6,7), or as a global density field (i.e. fig. 8). For convenience, periodic density fields used in this paper are represented by the density for a single period and referred as density or distribution profiles. The profiles are created using simple shapes functions to allow interactive authoring and fast evaluation. Figure 3 demonstrates the range of appearance produced by this control over the distribution of impulses using different periodic density profiles. Our formulation encompasses previous noise formulations and is thus able to produce various patterns, from irregular patterns (by using a random distribution, cf. figure 3 top) , to near-regular patterns(cf. figure 3, bottom).

Note that in spite of regularity in global appearance, this texture preserves randomness : impulses are still generated using a random distribution process: only the density of impulses varies spatially. We experienced that the shape of density profiles allows an easy and intuitive control of texture structure : structural alignments result in small irregular gaps or aligned variations in the pattern. Such insights can be used to visually estimate the elements positioning margin within an area and to recreate the density profile accordingly. The user can also edit kernels in real-time, getting an instant texture feedback while creating or modifying the pattern.

4 BY EXAMPLE PROCEDURAL TEXTURING

As shown in section 3, our spot noise model is well suited to manage some types of structural features. We have further shown that the structure can be represented either by the kernel itself or by the distribution of impulses (using non-uniform distributions). However analyzing the input texture to obtain both the distribution and the shape of the kernel is a difficult and challenging task: both are strongly linked and they can only hardly be decoupled. A solution consists in fixing one of the two. In this paper we propose to use a similar approach as for LRP noise: we extract complex kernels from examples that are then distributed over a regular grid.

First, we briefly present a summary of the LRP Noise by example approach introduced to process some types of structured textures.

Summary of by-example LRP Noise

By-example LRP Noise is based on a spectrum segmentation to extract the magnitudes and phases of structural features : the input spectrum is stratified according to energy levels, and then subdivided in sub-strata to compute local noises.

A stratum R corresponding to the highest energy area in the power spectrum is considered as defining the structure of the pattern. This region is seen as the frequencies "containing the structure" and is chosen by a tunable parameter $r \in [0;1]$ such that the proportion of total energy contained within R is r. The most important

structural features are preserved by fixing the phases and amplitudes over the corresponding frequencies. At the extremities, the resulting texture varies from a fully procedural ($r = 0$) to a copy of the original sample ($r = 1$). In practice and for weakly structured random textures, authors report a value of $r \approx 20\%$ as an efficient value for preserving both the structural features and the randomness of the pattern. Final noise (texture) computation is done by summing the noises approximating all energy strata :

$$n = n_R + \sum_S n_S \tag{6}$$

Two types of noises are thus considered : noises n_S relying completely on power spectrum and purely random phases (they keep storage requirements minimal), while the noise n_R has fixed phases and amplitudes.

One drawback is that a high amount of cosine waves are needed to accurately represent n_R, thus generating an important computational overhead. Gilet et al. [GSV*14] deal with this issue by trading continuity for computational efficiency. Basically, the structure image (the inverse Fourier transform of n_R) is iteratively decomposed into a regular grid of blocks. A block-wise FFT of this structure image is computed and a fixed amount of highest-amplitude frequencies are selected and stored for each block. n_R is finally evaluated by block in the spatial domain and re-assembled during rendering using the windowing function. We refer the reader to [GSV*14] for more details about LRP Noise by example.

4.1 Reproducing structure with Locally Controlled Spot Noise

First we have to separate the input texture into the structural part and the Gaussian random part. To this end, we propose to use the same technique as for the LRP Noise method, i.e. to consider a *structure* image constructed as the inverse Fourier transform of the highest energy region of the spectrum. The goal of our method is to compute a collection of I local kernels, each encoding a part of the structure. We achieve this by subdividing the structure image following a regular grid of arbitrary resolution and computing a Gaussian-based representation for each of the resulting blocks.

This ends up in computing a compact elliptical Gaussian-based representation of a given image, which is a difficult process when images are complex. By using a standard ellipse fitting algorithm, such as the method proposed in [AWF95], the pixels of the input image are approximated by J ellipses, which are then expressed as elliptical Gaussian functions of corresponding radius. As illustrated in figure 4, these functions are the basis of our local spot noise kernel. The efficiency of ellipse fitting is strongly depending on the complexity of the image and is able to work only on simple features such as presented in figures 4 and 5. A deeper analysis and segmentation of the image could further lead to an increase of the quality of the approximation and could in future work allow the reproduction of more complex structural features.

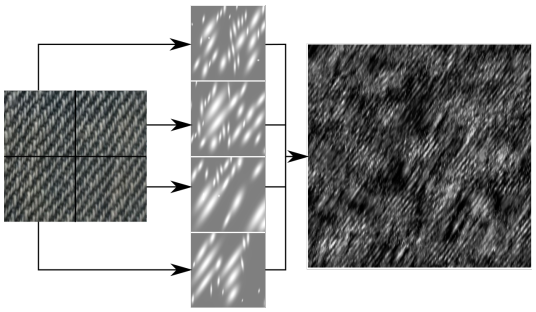


Figure 4: Noise by example : an input sample (left image) is subdivided into smaller samples. Each sub samples is decomposed in Gaussian functions (around 20 for this example), giving for each sub-samble a specific spot (center). Resulting pattern (right) is obtained by random distribution of the spots. Performances are about 10 fps.

The number J of Gaussian functions is constant for all blocks of the image and impact the accuracy of the approximation of each structural feature and the performance of the spot-noise during rendering. The rendering speed is linearly dependant on the number of Gaussian functions composing each kernel. This trade-off between accuracy and performance is a parameter of our model and chosen by the user. In practice, all results in this paper are computed with J between 4 and 8, and up to 20 for very complex spots.

4.2 Combining structure and noise

During rendering, the impulses are distributed using jittering (random displacement of points defined on the integer lattice). The resolution of the lattice corresponds to the resolution used during the subdivision process of the structure image. Each impulse is associated with a kernel, that can be chosen as the kernel approximating the block (in the structure image) corresponding to the integer lattice of the impulse or randomly chosen to increase randomness. By using the kernel approximating the corresponding structure block, low frequency structural features can be represented by the combination of kernel across the output, at the cost of the randomness.
The Gaussian random part of the input texture is then added by a cosine-based kernel noise as in the standard

LRP Noise method or as a random distribution of simple kernels defined as in figure 1.

5 RESULTS

We implemented our noise as a GPU fragment shader using OpenGL. Random numbers were generated by a linear congruential PRNG initialised by a Morton coded seed similarly to [LLDD09]. Performances are strongly dependant on the impulses density and kernels complexity. All results in this paper are rendered between 10 and 165 fps in a 1200×1200 window on a GeForce 980.

Figures 4 and 5 shows examples of structure reproduction obtained from an input example. As can be seen, the simple shape of the input structural feature is accurately reproduced by our automatic process. As stated earlier, our method relies on automatic segmentation and computation of Gaussian representation of an input pattern. Using a straightforward ellipse fitting technique provides results for simple patterns but automatic analysis of a complex pattern remains a difficult challenge. We however believe that this is a first step toward fully automatic by-example procedural texturing of complex patterns using locally controlled spot noises.

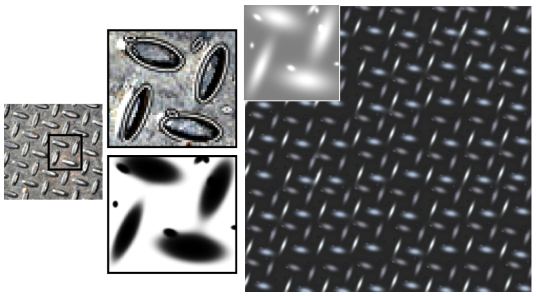


Figure 5: A simple example of pattern modeling obtained from the input example (left). The shape of the structural pattern is computed by ellipse fitting and expressed as a sum of Gaussian functions (middle) to produce the final structure (right). Performances are around 165 fps.

Several patterns can still be represented using an user provided kernel. Figure 6 shows several examples of pattern reproduction through a given periodic distribution and an adequate user-defined kernel.

Unlike recently introduced noise methods, our technique focus on the edition of pattern in the spatial domain. Indeed, interactive modeling in the spatial domain is easier than the direct edition of a power spectrum. Figure 7 illustrates our edition pipeline and shows how the kernel and distribution can be edited to impact the global structure of the target pattern with instant feedback for the user. Figure 8 illustrates the control capabilities of our noise model over multiple structures distribution within a single pattern. Figure 9 illustrates

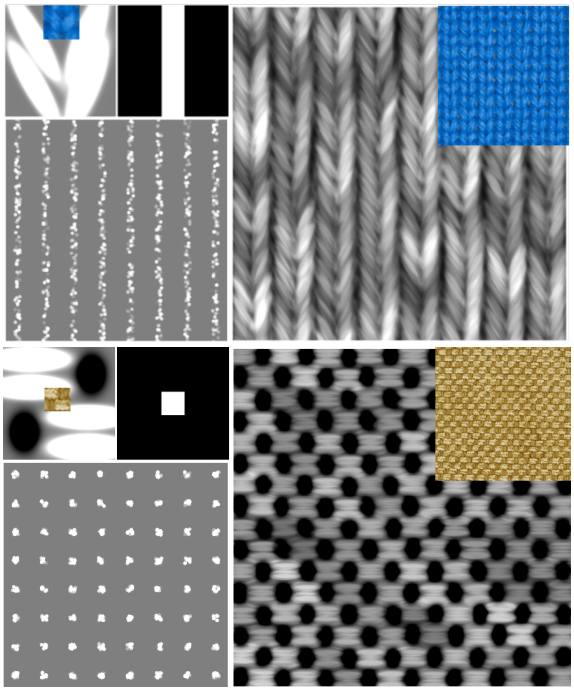


Figure 6: Examples of a near-regular features reproduction by a single spot noise. The kernel profile (left column,top left profile) is provided as a user-guided Gaussian decomposition of a texture element (i.e. a mesh of fabric). The distribution profile (left column, top right profile) is either directly authored or provided as a small texture. The top right tiles show the original pattern from which the sample to reproduce as a kernel were extracted. Performances are around 60 fps for the top pattern and 40 fps for the bottom pattern.

an application of a pattern on a 3D object with a simple bump mapping. This figure uses the top row configuration of the figure 6 to compute a noise used as a height field. Normals used for the bump mapping were computed by finite differences over 3 evaluations of the noise in a fragment shader.

6 CONCLUSION

We have introduced a new noise model based on locally controlled spot noises to reproduce from near-regular to irregular pattern features. Near-regular features are produced by combination of structured kernel and a controlled random distribution process. As it extends the Local Phase Noise model, it can still reproduce irregular patterns with structural features.

Our noise function contrasts with most recent research papers concerning noise models because our focus is not to match a given power spectrum, but rather to focus on spatial structure control : sculpting interactively a pattern shape in spatial domain is an easier creation process than editing a given power spectrum. Noise is hard to control, and generally ill suited for the modeling of structured procedural textures. We believe that

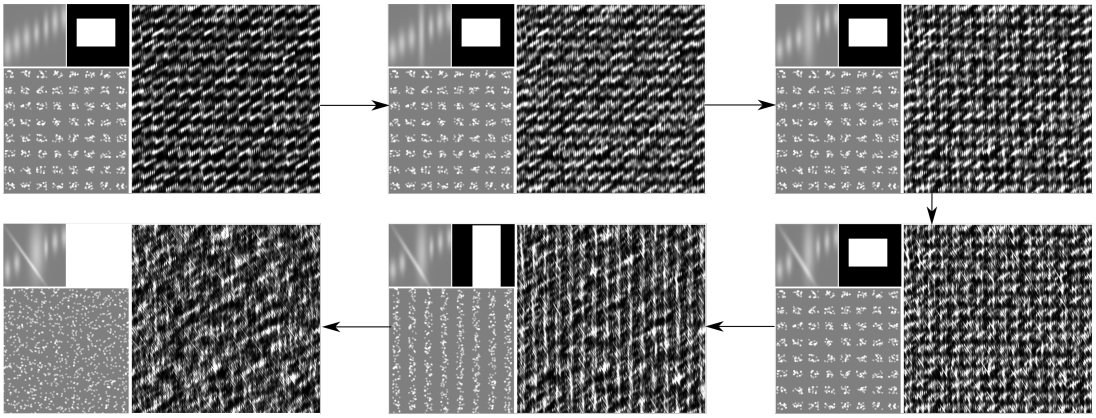


Figure 7: An example of edition process. Giving an input spot and distribution (top left), the user can interactively modify each Gaussian function of the spot and the distribution profile to change the appearance of the result. Performances are around 45 fps.

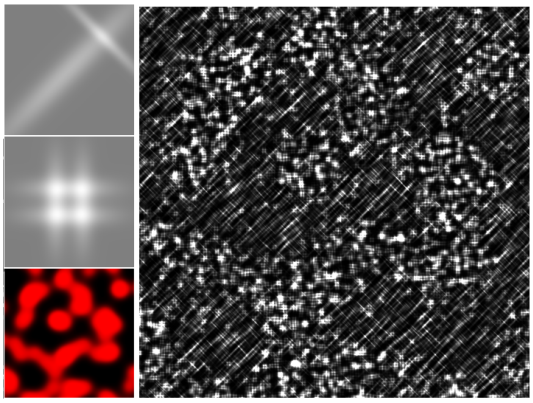


Figure 8: An example of mixed kernels for structures repartition within a single noise pattern. A global density field (bottom left), generated by a secondary spot noise, is used to control the distribution of spots : for an impulse distributed, the corresponding spot (top left profile or middle left profile) is selected according to a random density test. Performances are around 55 fps.

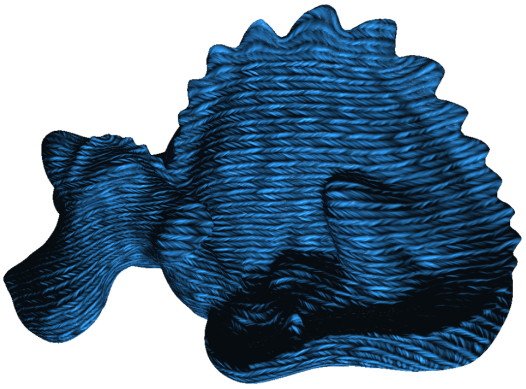


Figure 9: An example of the blue fabric pattern (fig. 6, top row) applied on a 3D model. The noise function was used to compute heights. Normals of the resulting pattern were computed using finite differences and used in a simple bump mapping process. Performances are around 20 fps.

our locally controlled spot noise could provide a first hint to address the difficult problem of modeling structured patterns. In particular, one important extension for future work would be the exploration of an automatic example-based method. Such a method would use as input a photograph of surface details and then attempt to derive a corresponding set of kernels and impulse density distributions.

7 REFERENCES

[AWF95] ANDREW W. FITZGIBBON R.: A buyer's guide to conic fitting. In *Proc. 5th British Machine Vision Conference, Birmingham* (1995), pp. 513–522.

[CJ83] CROSS G. R., JAIN A. K.: Markov random field texture models. *IEEE Trans. Pattern Anal. Mach. Intell. 5*, 1 (Jan. 1983), 25–39.

[CSHD03] COHEN M. F., SHADE J., HILLER S., DEUSSEN O.: Wang tiles for image and texture generation. *ACM Trans. Graph. 22*, 3 (July 2003), 287–294.

[dLvL97] DE LEEUW W., VAN LIERE R.: Divide and conquer spot noise. In *Proceedings of the 1997 ACM/IEEE Conference on Supercomputing* (New York, NY, USA, 1997), SC '97, ACM, pp. 1–13.

[EF01] EFROS A. A., FREEMAN W. T.: Image quilting for texture synthesis and transfer. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 2001), SIGGRAPH '01, ACM, pp. 341–346.

[EMP*02] EBERT D. S., MUSGRAVE F. K., PEACHEY D., PERLIN K., WORLEY S.: *Texturing and Modeling: A Procedural Approach*, 3rd ed. Morgan Kaufmann Publishers Inc., San

Francisco, CA, USA, 2002.

[GD10] GILET G., DISCHLER J. M.: Procedural texture particles. In *Proceedings of the 2010 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games* (New York, NY, USA, 2010), I3D '10, ACM, pp. 6:1–6:1.

[GDG12a] GILET G., DISCHLER J.-M., GHAZANFARPOUR D.: Multi-scale assemblage for procedural texturing. *Computer Graphics Forum 31*, 7 (2012), 2117–2126.

[GDG12b] GILET G., DISCHLER J.-M., GHAZANFARPOUR D.: Multiple kernels noise for improved procedural texturing. *The Visual Computer 28*, 6 (2012), 679–689.

[Gla04] GLANVILLE S.: Texture bombing. In *GPU Gems*. 2004, ch. 20.

[GLLD12] GALERNE B., LAGAE A., LEFEBVRE S., DRETTAKIS G.: Gabor noise by example. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2012) 31*, 4 (July 2012). To appear.

[GSV*14] GILET G., SAUVAGE B., VANHOEY K., DISCHLER J.-M., GHAZANFARPOUR D.: Local random-phase noise for procedural texturing. *ACM Trans. Graph. 33*, 6 (Nov. 2014), 195:1–195:11.

[JBL*06] JANG Y., BOTCHEN R. P., LAUSER A., EBERT D. S., GAITHER K. P., ERTL T.: Enhancing the Interactive Visualization of Procedurally Encoded Multifield Data with Ellipsoidal Basis Functions. *Computer Graphics Forum 25*, 3 (2006), 587–596.

[LD05] LAGAE A., DUTRÉ P.: A procedural object distribution function. *ACM Trans. Graph. 24*, 4 (Oct. 2005), 1442–1461.

[Lew89] LEWIS J. P.: Algorithms for solid noise synthesis. *SIGGRAPH Comput. Graph. 23*, 3 (July 1989), 263–270.

[LLC*10] LAGAE A., LEFEBVRE S., COOK R., DEROSE T., DRETTAKIS G., EBERT D., LEWIS J., PERLIN K., ZWICKER M.: A survey of procedural noise functions. *Computer Graphics Forum 29*, 8 (December 2010), 2579–2600.

[LLDD09] LAGAE A., LEFEBVRE S., DRETTAKIS G., DUTRÉ P.: Procedural noise using sparse gabor convolution. In *ACM SIGGRAPH 2009 papers* (New York, NY, USA, 2009), SIGGRAPH '09, ACM, pp. 54:1–54:10.

[LVLD10] LAGAE A., VANGORP P., LENAERTS T., DUTRÉ P.: Procedural isotropic stochastic textures by example. *Computers '&' Graphics (Special issue on Procedural Methods in Computer Graphics) 34*, 4 (2010), 312–321.

[OL81] OPPENHEIM A. V., LIM J. S.: The Importance of Phase in Signals. *Proceedings of the IEEE 69*, 5 (May 1981), 529–541.

[Per85] PERLIN K.: An image synthesizer. *SIGGRAPH Comput. Graph. 19*, 3 (July 1985), 287–296.

[VGR16] VERSTEEGEN R., GIMEL'FARB G., RIDDLE P.: Texture modelling with nested high-order markov-gibbs random fields. *Computer Vision and Image Understanding 143* (2016), 120 – 134. Inference and Learning of Graphical Models Theory and Applications in Computer Vision and Image Analysis.

[VSLD13] VANHOEY K., SAUVAGE B., LARUE F., DISCHLER J.-M.: On-the-fly multi-scale infinite texturing from example. *ACM Trans. Graph. 32*, 6 (Nov. 2013), 208:1–208:10.

[vW91] VAN WIJK J. J.: Spot noise texture synthesis for data visualization. *SIGGRAPH Comput. Graph. 25*, 4 (July 1991), 309–318.

[WM03] WELSH T., MUELLER K.: A frequency-sensitive point hierarchy for images and volumes. In *Visualization, 2003. VIS 2003. IEEE* (Oct 2003), pp. 425–432.

ISSN 2464-4617 (print) WSCG 2016 - 24th Conference on Computer Graphics, Visualization and Computer Vision 2016
ISSN 2464-4625 (CD-ROM)

Optical Flow Estimation via Steered-L^1 Norm

Ammar Zayouna
School of Science and Technology
Middlesex University
London NW4 4BT, UK
a.zayouna@mdx.ac.uk

Richard Comley
School of Science and Technology
Middlesex University
London NW4 4BT, UK
r.comley@mdx.ac.uk

Daming Shi
School of Science and Technology
Middlesex University
London NW4 4BT, UK
d.shi@mdx.ac.uk

ABSTRACT

Global variational methods for estimating optical flow are among the best performing methods due to the sub-pixel accuracy and the 'fill-in' effect they provide. The fill-in effect allows optical flow displacements to be estimated even in low and untextured areas of the image. The estimation of such displacements are induced by the smoothness term. The L^1 norm provides a robust regularisation term for the optical flow energy function with a very good performance for edge-preserving. However this norm suffers from several issues, among these is the isotropic nature of this norm which reduces the fill-in effect and eventually the accuracy of estimation in areas near motion boundaries. In this paper we propose an enhancement to the L^1 norm that improves the fill-in effect for this smoothness term. In order to do this we analyse the structure tensor matrix and use its eigenvectors to steer the smoothness term into components that are 'orthogonal to' and 'aligned with' image structures. This is done in primal-dual formulation. Results show a reduced end-point error and improved accuracy compared to the conventional L^1 norm.

Keywords

Optical flow, Variational methods, $TV-L^1$, Structure tensor.

1 INTRODUCTION

Optical flow is an important cue in image processing applications. It can be defined as the estimation of image point displacements over time [1], [2]. Such images are taken for the same scene at successive moments in time. Methods for finding optical flow can be classified in many ways. An early classification can be found in the work of Barron et al. [3], which classified optical flow algorithms into four main groups. One of these groups relies on the computation of optical flow using the calculus of variations and is thus denoted as 'Variational methods'. Variational methods for estimating optical flow belong to the highest-accuracy methods. These methods find the optical flow displacement field by minimising an energy function mainly comprising data and smoothness terms:

$$E = \alpha E_{data} + E_{smooth} \tag{1}$$

where α controls the weight between the two terms. The data term E_{data} is based on the brightness constancy assumption, where it is assumed that illumination between images does not change over time:

$$I_2(\mathbf{x}+\mathbf{u}, t+1) = I_1(\mathbf{x}, t)$$

where $\mathbf{u} = (u, v)$ is the displacement for each pixel in the x and y directions respectively, $\mathbf{x} = (x, y)$ is the pixel coordinates, and $t \in [0, T]$ is the time reference. This term is linearised using the 'Taylor Expansion' to obtain what is known as the 'optical flow constraint' [4]:

$$I_x u + I_y v + I_t = 0 \tag{2}$$

where subscripts denote partial derivatives. The brightness constancy assumption does not always hold, as it gets violated when illumination changes between the two images, for example due to shadows and shading. The smoothness (or regularity term) on the other hand is based on the spatial constancy assumption, where it is assumed that the neighbouring pixels in the first image are still neighbours in the second image. Hence diverse displacements are penalised. This assumption also does not always hold as it gets violated in some areas such as motion boundaries, where pixels in the first image are no longer neighbours in the second image due to motion or occlusion. One example of an optical flow energy function can be found in the early work of Horn-Schunck [1], where they proposed to minimise the following energy function:

$$E = \int_\Omega \big(\overbrace{\alpha (I_x u + I_y v + I_t)^2}^{E_{data}} + \overbrace{|\nabla u, \nabla v|^2}^{E_{smooth}} \big) dxdy \tag{3}$$

where Ω is a 2D image domain and $\nabla u = (u_x, u_y)$ and $\nabla v = (v_x, v_y)$. The solution is found using the Euler-Lagrange equations, which results in a couple of simultaneous equations. The displacements then can be

easily found by solving the resulting system of equations. In that algorithm a quadratic norm is used in the smoothness term. The quadratic norm penalises the flow field severely in all directions, hence it produces blurry motion edges due to this penalisation.

The displacement field is piecewise in nature, hence a good choice for a smoothness term is a piecewise function that characterises the piecewise nature of such a displacement field. The total variation L^1 is an example of such functions. Indeed this piecewise function can characterise the flow field efficiently. However, this norm suffers from two main issues. First, it is not continuously differentiable. This issue was addressed by Chambolle [5]. Later Zach et al. [6] used this solution to propose an optical flow estimation algorithm in a primal-dual formulation. Primal-dual algorithms, in addition to their accuracy and good edge-preserving qualities (due to the use of the L^1 norm in the smoothness term), can be easily parallelised using modern graphics hardware [7], [8], [9].

The second issue is that the L^1 norm is isotropic, hence the fill-in effect [10] reduces along motion boundaries. In this paper we propose to improve L^1 by the use of eigenvectors of the local structure tensor. We derive the formulation for this in the primal-dual settings. The eigenvectors of the structure tensor were used by Zimmer et al. [11] to improve the fill-in effect using robust functions which approximate the behaviour of the L^1 norm as a smoothness term. In the current work we apply it directly to the total variation L^1 norm, which is non-trivial due to the non-continuous differentiability of the L^1 norm. Additionally we use a data term inspired by the delayed linearisation data term proposed by Brox et al. [2].

This paper is organised as follows, Section-2 examines some related work. Section-3 includes a brief introduction for the notion of 'Structure tensor'. Section-4 introduces the method. Section-5 discusses the implementation and results. Section-6 concludes this paper and proposes several enhancements to be investigated in the future.

2 RELATED WORK

Since the marquee work of Horn-Schunck [1], a lot of research has been dedicated to improve the estimation of global optical flow algorithms. Both the data and the smoothness terms have undergone a lot of improvements. The Horn-Schunck method belongs to what is known as the 'Global methods'. Lucas-Kanade [12] proposed to calculate optical flow by assuming that the displacement field is constant in a small local neighbourhood, hence this type of method was called 'Local methods'. Bruhn et al. [4] proposed to combine the global and local methods by integrating neighbouring pixels in the data term using a Gaussian filter kernel, in what is known as the Combined Local-Global (CLG) method. Brox et al. [2] proposed to delay linearisation of the data term of the optical flow equation. This enabled the computation of high accuracy displacement fields. Additionally to improve robustness several algorithms extended the data term to include image gradients, thus improving the robustness to illumination changes [2], [13], [11]. Wedel et al. [8] proposed to improve the robustness to illumination changes by introducing a structure-texture decomposition step before the minimisation.

In the smoothness term, Horn-Schunck [1] used a quadratic function as a regularisation term. The quadratic function penalises the flow field severely regardless of the flow magnitude, thus introducing blurriness across motion boundaries. To remedy this, several methods used robust functions in the smoothness term such as the Lorentzian function [14], the charbonneir [4] and the robust L^1 approximation function [4]. The total variation L^1 norm was also used as a smoothness term. The problem with the L^1 norm is that it is not continuously differentiable. Chambolle [5] proposed a numerical scheme to solve the $TV - L^1$ minimisation and applied it to image denoising and zooming. Zach el al. [6] used this scheme under primal-dual formulation minimisation to estimate optical flow. Drulea et al. [9] used this scheme to find the optical flow field and used a CLG data term. In addition to that [9] used a diffusion tensor [15] to improve the fill-in effect in low and untextured areas. However a drawback of using a diffusion tensor is that it produces over-segmentation, this is because this diffusion tensor is a function of image gradients. To improve the fill-in effect, Sun et al. [16] and later Zimmer et al. [11] analysed the image structure tensor to obtain eigenvectors, and used these eigenvectors to improve the fill-in effect. Hence the direction of penalisation is adapted to the direction of the local image structure, while the magnitude of this penalisation depends on the flow field magnitude.

Despite the recent advances in estimating optical flow fields using methods that are not variational, the variational methods are still needed. Probabilistic methods for example, despite their popularity, lack the sub-pixel accuracy of the variational methods. Hence variational methods are generally used as a final stage to refine the estimation of the displacement field [17], [18], [19], [20].

3 STRUCTURE TENSOR AND STEERING IMAGE DERIVATIVES

The structure tensor of a 2D image is a 2×2 matrix that contains information about the structure orientation

in a certain neighbourhood in that image. The initial structure tensor J_0 can be expressed as follows [21]:

$$J_0 = \nabla I \nabla I^T = \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

where I is a 2D image, ∇I is the image gradients $(I_x, I_y)^\mathsf{T}$ in the x, and y directions. The structure tensor J is obtained by integrating information in a certain neighbourhood. This is done by convolving the initial structure tensor J_0 with a Gaussian filter kernel G_ρ, where ρ is the standard deviation. The structure tensor is expressed as follows:

$$J = G_\rho * J_0 \tag{4}$$

where '$*$' denotes convolution. This structure tensor can be analysed and two orthonormal eigenvectors with corresponding eigenvalues are obtained. The first eigenvector can be formulated as $(\cos\phi, \sin\phi)$ and the second as $(-\sin\phi, \cos\phi)$ [22]. The eigensystem obtained can be used to give information about the local image structures. The eigensystem can be written in the following form:

$$J\mathbf{e} = \lambda\mathbf{e}$$

where $\mathbf{e}$ are the set of eigenvectors ($\mathbf{e}_1$, $\mathbf{e}_2$...$\mathbf{e}_n$), and λ are the corresponding eigenvalues (λ_1, λ_2,... λ_n), and n is the size of the square matrix J.

The first eigenvector corresponds to the largest eigenvalue points across the dominant structure in the neighbourhood, while the second eigenvector corresponds to the smaller eigenvalue points along that structure. Figure-1 depicts two eignevectors obtained by analysing the structure tensor at a certain location. The first eigenvector is depicted here in green, while the second eigenvector is depicted in red. This image is obtained from the Middlebury dataset [23].

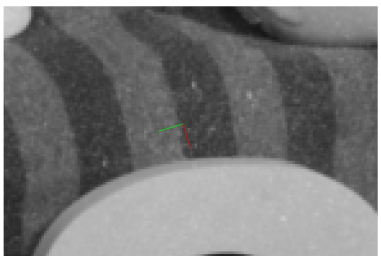


Figure 1: Eigenvectors directions of a structure tensor. The green line corresponds to the first eigenvector pointing across area with high gradient, while the red line corresponds to the second eigenvector pointing along area with high gradient.

The eigenvectors can be used to obtain what is known as the 'Steered image derivatives', where image gradients are steered from the conventional x and y direction to directions 'orthogonal to', and 'aligned with' the local image structures [22], [24]. This can be written in the following form:

$$I_o = \cos\phi . I_x + \sin\phi . I_y \tag{5}$$

$$I_a = -\sin\phi . I_x + \cos\phi . I_y \tag{6}$$

where ϕ is the angle of the first eigenvector and it is obtained via the structure tensor matrix.

4 STEERED–L^1 OPTICAL FLOW

It is desired in this paper to use the robust L^1 norm known for its edge preserving performance in the computation of an optical flow field. To this end it is required to minimise the following equation:

$$E = \int_\Omega \Big(\alpha E_{data}(I_1, I_2, \mathbf{u}) + |\nabla \mathbf{u}| \Big) dxdy \tag{7}$$

where $E_{data}(I_1, I_2, \mathbf{u})$ will be introduced later. The minimisation of this equation is non-trivial since the smoothness term used (the L^1 norm) is not continuously differentiable. Hence, following the primal-dual formulation [6], an auxiliary variable $\bar{\mathbf{u}}$ is introduced and the energy function takes the following form[1]:

$$E = \int_\Omega \Big(\alpha E_{data}(I_1, I_2, \mathbf{u}) + \frac{1}{2\theta}(\mathbf{u} - \bar{\mathbf{u}})^2 + |\nabla \mathbf{u}| \Big) \tag{8}$$

where θ is a small constant. This equation is split into a dual and a primal equation, the dual equation is written as follows:

$$E_{dual} = \int_\Omega \Big(\frac{1}{2\theta}(\mathbf{u} - \bar{\mathbf{u}})^2 + |\nabla \mathbf{u}| \Big) \tag{9}$$

and the primal equation is written as:

$$E_{primal} = \int_\Omega \Big(\alpha E_{data}(I_1, I_2, \mathbf{u}) + \frac{1}{2\theta}(\mathbf{u} - \bar{\mathbf{u}})^2 \Big) \tag{10}$$

The minimisation of this system of equations is performed in primal-dual steps. In the following subsections we discuss the primal and the dual steps in detail.

4.1 The Dual Step

The aim of the dual step is to minimise $\mathbf{u}$ while keeping $\bar{\mathbf{u}}$ fixed. We propose to steer the derivatives of the displacement fields according to the local structure. Hence Equation-9 is written as follows:

$$E_{dual} = \int_\Omega \Big(|\mathbf{e}^\mathsf{T} \nabla u| + \frac{1}{2\theta}(u - \bar{u})^2 + |\mathbf{e}^\mathsf{T} \nabla v| + \frac{1}{2\theta}(v - \bar{v})^2 \Big) \tag{11}$$

[1] Starting from this point, the notation '*dxdy*' is omitted for brevity.

where:

$$\mathbf{e}^{\mathsf{T}} = \begin{bmatrix} \cos\phi & \sin\phi \\ -\sin\phi & \cos\phi \end{bmatrix}$$

are the eigenvectors of the structure tensor, and $\nabla u, \nabla v$ are expressed here as $(u_x, u_y)^{\mathsf{T}}$ and $(v_x, v_y)^{\mathsf{T}}$. In this way the directions of the smoothness term is adapted to the direction of the local image structure, while the magnitude of the penalisation is adapted to the flow field itself [11].

To solve the minimisation, Euler-Lagrange equations are obtained [9]. The first equation obtained is the following:

$$-div\big(\mathbf{e}^{\mathsf{T}}.\frac{\nabla u}{|\nabla u|}\big) + \frac{1}{\theta}(u-\bar{u}) = 0 \tag{12}$$

which can be re-written as follows:

$$u = \theta.div(\mathbf{e}^{\mathsf{T}}.\mathbf{p}_u) + \bar{u} \tag{13}$$

where $\mathbf{p}_u = \frac{\nabla u}{|\nabla u|}$, it follows that:

$$\mathbf{p}_u.|\nabla u| - \nabla u = 0, \quad |\mathbf{p}_u| \leq 1 \tag{14}$$

Substituting Equation-13 in Equation-14, the following equation is obtained:

$$\mathbf{p}_u.\Big|\nabla\big(div(\mathbf{e}^{\mathsf{T}}.\mathbf{p}_u) + \bar{u}/\theta\big)\Big| - \nabla\big(div(\mathbf{e}^{\mathsf{T}}.\mathbf{p}_u) + \bar{u}/\theta\big) = 0.$$

Adding $\mathbf{p}_u$ to both sides of the above equation yields the following fixed-point iteration to find $\mathbf{p}_u$:

$$\mathbf{p}_u^{k+1} = \frac{\mathbf{p}_u^k + \tau.\nabla\Big(div(\mathbf{e}^{\mathsf{T}}.\mathbf{p}_u^k) + \bar{u}/\theta\Big)}{1 + \tau.\Big|\nabla\big(div(\mathbf{e}^{\mathsf{T}}.\mathbf{p}_u^k) + \bar{u}/\theta\big)\Big|} \tag{15}$$

where k is the iteration count, and τ is the step size. In the same way $\mathbf{p}_v$ can be obtained, and it is calculated using the following fixed-point iteration:

$$\mathbf{p}_v^{k+1} = \frac{\mathbf{p}_v^k + \tau.\nabla\Big(div(\mathbf{e}^{\mathsf{T}}.\mathbf{p}_v^k) + \bar{v}/\theta\Big)}{1 + \tau.\Big|\nabla\big(div(\mathbf{e}^{\mathsf{T}}.\mathbf{p}_v^k) + \bar{v}/\theta\big)\Big|}. \tag{16}$$

The terms $\mathbf{e}^{\mathsf{T}}.\mathbf{p}_u^k$, and $\mathbf{e}^{\mathsf{T}}.\mathbf{p}_v^k$ can be replaced by the alternative notations $\mathbf{p}_{su}$ and $\mathbf{p}_{sv}$, where:

$$\mathbf{p}_{su} = \mathbf{e}^{\mathsf{T}}.[p_{1u},\ p_{2u}]^{\mathsf{T}} \tag{17}$$

$$\mathbf{p}_{sv} = \mathbf{e}^{\mathsf{T}}.[p_{1v},\ p_{2v}]^{\mathsf{T}} \tag{18}$$

Hence, Equation-15 and Equation-16 can be written in the following way:

$$\mathbf{p}_u^{k+1} = \frac{\mathbf{p}_u^k + \tau.\nabla\Big(div\ \mathbf{p}_{su} + \bar{u}/\theta\Big)}{1 + \tau.\Big|\nabla\big(div\ \mathbf{p}_{su} + \bar{u}/\theta\big)\Big|}. \tag{19}$$

$$\mathbf{p}_v^{k+1} = \frac{\mathbf{p}_v^k + \tau.\nabla\Big(div\ \mathbf{p}_{sv} + \bar{v}/\theta\Big)}{1 + \tau.\Big|\nabla\big(div\ \mathbf{p}_{sv} + \bar{v}/\theta\big)\Big|}. \tag{20}$$

4.2 The Primal Step

The aim of the primal step is to minimise $\bar{\mathbf{u}}$ while keeping $\mathbf{u}$ fixed. Starting from a non-linearised data term, we propose to minimise the following data term:

$$E_{Primal} = \int_\Omega \Big(\alpha|I_2(\mathbf{x}+\mathbf{u}) - I_1(\mathbf{x})|^2 + \gamma|\nabla I_2(\mathbf{x}+\mathbf{u}) - \nabla I_1(\mathbf{x})|^2 + \frac{1}{2\theta}(\mathbf{u}-\bar{\mathbf{u}})^2\Big) \tag{21}$$

where α here is the weight of the data term, and γ is the weight of the image gradient term which is added here to improve the robustness to illumination changes. This robustness can be further improved using the robust function $\Psi(s^2) = \sqrt{s^2+\varepsilon^2}$, where ε is a small constant.

$$E_{Primal} = \int_\Omega \Big(\alpha\ \Psi\big(|I_2(\mathbf{x}+\mathbf{u}) - I_1(\mathbf{x})|^2\big) + \gamma\ \Psi\big(|\nabla I_2(\mathbf{x}+\mathbf{u}) - \nabla I_1(\mathbf{x})|^2\big) + \frac{1}{2\theta}(\mathbf{u}-\bar{\mathbf{u}})^2\Big). \tag{22}$$

The second image $I_2(\mathbf{x}+\mathbf{u})$ can be written in the following form using the Taylor Expansion:

$$I_2(\mathbf{x}+\mathbf{u}) = I_2(\mathbf{x}+\mathbf{u_0}) + (\mathbf{u}-\mathbf{u_0})\nabla I_2(\mathbf{x}+\mathbf{u_0}) \tag{23}$$

where $\mathbf{u_0} = (u_0, v_0)$ is the initial displacement of the flow field. Similarly image gradients can be linearised, and the image gradients term can be rewritten as follows:

$$I_{2x}(\mathbf{x}+\mathbf{u}) = I_{2x}(\mathbf{x}+\mathbf{u_0}) + (\mathbf{u}-\mathbf{u_0})\nabla I_{2x}(\mathbf{x}+\mathbf{u_0}). \tag{24}$$

$$I_{2y}(\mathbf{x}+\mathbf{u}) = I_{2y}(\mathbf{x}+\mathbf{u_0}) + (\mathbf{u}-\mathbf{u_0})\nabla I_{2y}(\mathbf{x}+\mathbf{u_0}). \tag{25}$$

Plugging all these terms together yield the following equation:

$$E_{primal} = \int_\Omega \alpha\ \Psi\Big(\big|I_{t0} + (\mathbf{u}-\mathbf{u_0})\nabla I_2\big|^2\Big) + \gamma\ \Psi\Big(\big|\big(I_{tx} + (\mathbf{u}-\mathbf{u_0})\nabla I_{2x}\big), \big(I_{ty} + (\mathbf{u}-\mathbf{u_0})\nabla I_{2y}\big)\big|^2\Big)\Big) + \frac{1}{2\theta}(\mathbf{u}-\bar{\mathbf{u}})^2 \tag{26}$$

where the notation $(\mathbf{x}+\mathbf{u_0})$ was omitted from $\nabla I_2, \nabla I_{2x}, \nabla I_{2y}$ for brevity. The solution requires the minimisation of $E(\bar{\mathbf{u}})$. Hence Equation-26 is written as follows:

$$E_{primal} = \int_\Omega \alpha\ \Psi\Big(\big|I_{t0} + (\bar{\mathbf{u}}-\mathbf{u_0})\nabla I_2\big|^2\Big) + \gamma\ \Psi\Big(\big|\big(I_{tx} + (\bar{\mathbf{u}}-\mathbf{u_0})\nabla I_{2x}\big), \big(I_{ty} + (\bar{\mathbf{u}}-\mathbf{u_0})\nabla I_{2y}\big)\big|^2\Big)\Big) + \frac{1}{2\theta}(\mathbf{u}-\bar{\mathbf{v}})^2 \tag{27}$$

with the following abbreviation used:

$$\begin{aligned} I_{t0} &= I_2(\mathbf{x}+\mathbf{u_0}) - I_1(\mathbf{x}) \\ I_{tx} &= \frac{\partial}{\partial x} I_2(\mathbf{x}+\mathbf{u_0}) - \frac{\partial}{\partial x} I_1(\mathbf{x}) \\ I_{ty} &= \frac{\partial}{\partial y} I_2(\mathbf{x}+\mathbf{u_0}) - \frac{\partial}{\partial y} I_1(\mathbf{x}) \end{aligned} \tag{28}$$

The minimisation of Equation-27 can be easily performed by setting the derivatives of the equation with respect to $\bar{u}, \bar{v}$ equal to zero. This leads to the following set of equations:

$$\begin{aligned}&\big[\alpha\,\Psi_1^{'}.I_{2x}^2+\gamma\,\Psi_2^{'}(I_{2xx}^2+I_{2yx}^2)+\frac{1}{\theta}\big]\bar{u}\\&+\big[\alpha\,\Psi_1^{'}.I_{2x}I_{2y}+\gamma\,\Psi_2^{'}.I_{2xy}(I_{2xx}+I_{2yy})\big]\bar{v}\\&=-\big[\alpha\,\Psi_1^{'}r_{t0}I_{2x}+\gamma\,\Psi_2^{'}r_{tx0}I_{2xx}+\gamma\,\Psi_2^{'}r_{ty0}I_{2xy}\big]+\frac{u}{\theta}.\end{aligned} \tag{29}$$

Similarly derivation with respect to $\bar{v}$ yields the following equation:

$$\begin{aligned}&\big[\alpha\,\Psi_1^{'}.I_{2x}I_{2y}+\gamma\,\Psi_2^{'}I_{2yx}(I_{2xx}+I_{2yx})\big]\bar{u}\\&+\big[\alpha\,\Psi_1^{'}.I_{2y}^2+\gamma\,\Psi_2^{'}.(I_{2xy}^2+I_{2yy}^2)\big]\bar{v}\\&=-\big[\alpha\,\Psi_1^{'}r_{t0}I_{2y}+\gamma\,\Psi_2^{'}r_{tx0}I_{2xy}+\gamma\,\Psi_2^{'}r_{ty0}I_{2yy}\big]+\frac{v}{\theta}\end{aligned} \tag{30}$$

where $\Psi^{'}$ is the derivative of Ψ. Ψ_1 and Ψ_2 are defined as follows:

$$\Psi_1=\Psi\Big(\big|I_{t0}+(\bar{\mathbf{u}}-\mathbf{u_0})\nabla I_2\big|^2\Big)$$

$$\Psi_2=\Psi\big(\big|\big(I_{tx}+(\bar{\mathbf{u}}-\mathbf{u_0})\nabla I_{2x}\big),\big(I_{ty}+(\bar{\mathbf{u}}-\mathbf{u_0})\nabla I_{2y}\big)\big|^2\big)$$

The values of $r_{t0},\ r_{tx0},\ r_{ty0}$ are given as follows:

$$\begin{aligned}r_{t0}&=I_{t0}-u_0I_{2x}-v_0I_{2y}\\r_{tx0}&=I_{tx}-u_0I_{2xx}-v_0I_{2xy}\\r_{ty0}&=I_{ty}-u_0I_{2xy}-v_0I_{2yy}\end{aligned} \tag{31}$$

4.3 Colour Image Realisation

The algorithm discussed so far can work on grey-scale images. Colour images offer richer photometric information compared to gray-scale images [25], [26]. It is possible to extend this algorithm to work with colour images. In RGB images, which is an additive colour model, colour is encoded in three channels (Red, Green and Blue). In order to be able to extend the work in our algorithm to colour images, the primal step is extended to incorporate the three colour channels. Hence Equation-29 and Equation-30 are written in the following form:

$$\begin{aligned}&\big[\alpha\,(\Psi_1^c)^{'}.(I_{2x}^c)^2+\gamma\,(\Psi_2^c)^{'}((I_{2xx}^c)^2+(I_{2yx}^c)^2)+\frac{1}{\theta}\big]\bar{u}^c\\&+\big[\alpha\,(\Psi_1^c)^{'}.I_{2x}^cI_{2y}^c+\gamma\,(\Psi_2^c)^{'}.I_{2xy}^c(I_{2xx}^c+I_{2yy}^c)\big]\bar{v}^c\\&=-\big[\alpha\,(\Psi_1^c)^{'}r_{t0}^cI_{2x}^c+\gamma\,(\Psi_2^c)^{'}r_{tx0}^cI_{2xx}^c+\gamma\,(\Psi_2^c)^{'}r_{ty0}^cI_{2xy}^c\big]+\big(\frac{u}{\theta}\big)^c\end{aligned} \tag{32}$$

$$\begin{aligned}&\big[\alpha\,(\Psi_1^c)^{'}.I_{2x}^cI_{2y}^c+\gamma\,(\Psi_2^c)^{'}I_{2yx}^c(I_{2xx}^c+I_{2yx}^c)\big]\bar{u}^c\\&+\big[\alpha\,(\Psi_1^c)^{'}.(I_{2y}^c)^2+\gamma\,(\Psi_2^c)^{'}.((I_{2xy}^c)^2+(I_{2yy}^c)^2)\big]\bar{v}^c\\&=-\big[\alpha\,(\Psi_1^c)^{'}r_{t0}^cI_{2y}^c+\gamma\,(\Psi_2^c)^{'}r_{tx0}^cI_{2xy}^c+\gamma\,(\Psi_2^c)^{'}r_{ty0}^cI_{2yy}^c\big]+\big(\frac{v}{\theta}\big)^c\end{aligned} \tag{33}$$

where $c\in\{c_1,c_2,c_3\}$ are the three colour channels in the RGB colour model. The values of $\mathbf{u}$ and $\bar{\mathbf{u}}$ are replicated at each iteration to cope with the three colour channels, and thus to obtain $\mathbf{u}^c$ and $\bar{\mathbf{u}}^c$. Additionally the values of $\mathbf{u}^c$ and $\bar{\mathbf{u}}^c$ are averaged before starting the dual step (see Algorithm-1).

4.4 Extended Intermediate filtering

Median filtering is used in optical flow algorithms to improve the computation of the displacement fields. The use of median filtering was found especially to be useful in the algorithms following the primal-dual formulation [8]. A Median filter is applied in each warp to the estimated flow field u, v to remove outliers. Median filters work by replacing the value of a certain pixel with the median pixel value in a certain neighbourhood. One can say that the use of median filters in this case encourages smoother solution (i.e. without outliers) in the estimated flow field. In this context we propose to extend the intermediate filtering by adding another filtering step. In the intermediate filtering stage we opt to use a bi-lateral filter [27] in addition to the median filter. Bi-lateral filters are known to have a good edge preserving performance. Unlike the Gaussian filter, the bi-lateral filter changes weight according to spatial distance and the colour (or intensity) difference.

$$I_f(x_i)=\frac{1}{K}\sum_{x\in\Omega_n}g(x_i-x)s(I(x_i)-I(x)).I(x) \tag{34}$$

where I is the original image, I_f is the filtered version of the image, K is a normalising term, Ω_n is the neighbourhood region, $g(x_i-x)$ is the kernel determining the weight based on spatial distance (which can be Gaussian), and $s(I(x_i)-I(x))$ is the kernel determining the weight based on colour difference.

The intermediate filtering proposed here is a two stage filtering that includes both the median and bi-lateral filters (see Algorithm-1). We call the new intermediate filtering 'Extended Intermediate Filtering' (EIF). Applying this filtering in each warp was found to improve the accuracy of the optical flow computation.

5 EXPERIMENTS

5.1 Implementation

The algorithm is written in MATLAB. Since this algorithm is variational, it can only detect small displacements. Hence the minimisation is performed in a Coarse-to-Fine (C2F) framework. To this end, the image sequence is downscaled several times to obtain a pyramid of images. The optical flow is first found in the coarsest version of the image sequence, the estimated flow is then propagated to the next finer layer and used as an initialisation for the solution in that layer. At

each layer, the second image is warped towards the first image using the flow estimated in the previous pyramid layer. A fine resolution pyramid is chosen for the minimisation [2] with 80 layers and downscaling ratio[2] of 0.95. Image and flow field resize is performed via bicubic interpolation. Image gradients are obtained via the kernel $[-1,\ 9,\ -45,\ 0,\ 45,\ -9,\ 1]/(60)$ [4]. The divergence and derivative for the variable $\mathbf{p}$ are approximated using the three point kernel $[-1,\ 0,\ 1]$. At each layer the second image and its derivatives are warped six times. At each layer the structure tensor is computed, with image derivatives computed using a 5×5 optimised derivative filter D [22]:

$$D = (0.0234, 0.2415, 0.4700, 0.2415, 0.0234)^{\mathsf{T}} * (0.0838, 0.3323, 0, -0.3323, -0.0838) \quad (35)$$

After calculating the structure tensor, eigen-decomposition is performed to find the two eigenvectors. To obtain $(div\ \mathbf{p}_{su})$ and $(div\ \mathbf{p}_{sv})$ the following filter kernels are used [29]:

$$h_x = \frac{1}{32}\begin{bmatrix} 3 & 0 & -3 \\ 10 & 0 & -10 \\ 3 & 0 & -3 \end{bmatrix} \quad (36)$$

$$h_y = \frac{1}{32}\begin{bmatrix} -3 & 10 & -3 \\ 0 & 0 & 0 \\ 3 & 10 & 3 \end{bmatrix} \quad (37)$$

The minimisation of the aforementioned formulation is done in primal-dual and C2F frameworks [6]. The parameters of the algorithm were set to the following values ($\alpha = 1/4700, \gamma = 1, \tau = 1/10, \theta = 1/10$, $\varepsilon = 0.001$). The minimisation pseudo-code is depicted in Algorithm-1.

Algorithm 1: Implementation Algorithm of steered-L^1 norm.

Input: Images I_1 & I_2,
number of pyramid layers $L = 80$, current layer l
Create pyramid of images with L layer, and a downscale ratio of 0.95;
initialization;
l=1;
initialise (u_l, v_l) to $(\mathbf{0}, \mathbf{0})$;
while $l \leq L$ **do**
  Up-scale size of (u_{l-1}, v_{l-1}) to (u_l, v_l);
  Find eigenvectors of the structure tensor of I_1;
  while *No. of Warps* $\leq$ *6* **do**
    Warp I_{2l}, I_{2x}, I_{2y} towards I_{1l}, I_{1x}, I_{1y};
    while *No. of iterations* $\leq$ *20* **do**
      Replicate the terms and update the auxiliary variable $\bar{u}^c, \bar{v}^c$ by solving the simultaneous equations (Equatin-32, 33) ;
      Average $\bar{u}^c, \bar{v}^c$ to yield $\bar{u}, \bar{v}$;
      Update u_l using Equation-13, and similarly update v_l;
      Calculate p_{su}, p_{sv} via (Equation-17, 18);
      Update p_u, p_v (Equation-19, 20);
    Apply median filter to u_l, v_l;
    Apply bi-later filter to u_l, v_l;

Output: (Displacement field (u, v))

5.2 Results

Several datasets are available to use for assessment of optical flow methods [23], [30]. In this section we use some of these datasets to assess the work of our algorithm. We highlight the improvement that the steered-L^1 norm introduces over the use of the L^1 norm.

5.2.1 Middlebury Dataset

The Middlebury dataset has been used in assessing the performance of optical flow algorithms for many years [23]. It contains synthetic and non-synthetic image sequences, and it includes some image sequences with known ground truth which can be used for training. To examine the performance differences that the steered-L^1 has made to the accuracy of optical flow estimation, optical flow displacement field is computed for the eight sequences that have a known ground truth, and the Average End-Point Error (AEPE) [23] is computed for these sequences. Table-1 depicts the difference in AEPE using the two norms. Table-1 shows clearly improved results obtained via using the steered-L^1 norm as the smoothness term.

Our method currently has an average rank of (57.5) on the AEPE Middlebury benchmark ranking table, and an average rank of (57.8) on the Average Angular Error AAE table. Figure-2 depicts a segment of the AEPE ranking table[3].

To further asses the performance of the proposed algorithm, we compare the results of this method with other methods sharing similar principals in the Middlebury ranking table. The first method to compare with is the improved $TV - L^1$ algorithm [8]. This algorithm follows a similar minimisation framework in a primal-dual formulation. However, it differs in the data term where a structure-texture decomposition is used to improve the robustness to illumination changes. In addition to that, there is a difference in the smoothness term where we steer it in accordance with the local structure. The second algorithm is the Large Displacement Optical Flow LDOF [28]. The third method to compare with is the CLG-TV [9], where the authors use a combined local-global data term (CLG). Additionally, the authors in this paper use an anisotropic diffusion filter to improve the fill-in effect. Table-2 illustrates the

[2] In general the number of layers in the pyramid can be chosen such that the discrete derivative filter kernel can be applied at the coarsest layer [28]. However 80 layers was enough to give good results in our experiments.

[3] `http://vision.middlebury.edu/flow/eval/`

	RubberWhale	Dimetrodon	Urban2	Urban3	Venus	Grove2	Grove3	Hydrangea	Average
L^1	0.09	0.15	0.55	0.49	0.32	0.18	0.57	0.17	0.32
Steered-L^1	0.08	0.14	0.53	0.46	0.31	0.17	0.57	0.16	0.30

Table 1: AEPE for colour images of the Middlebury dataset depicting the difference between the total variation L^1 and the steered-L^1 smoothness terms.

Average endpoint error	avg. rank	Army (Hidden texture) GT im0 im1 all	disc	untext	Mequon (Hidden texture) GT im0 im1 all	disc	untext	Schefflera (Hidden texture) GT im0 im1 all	disc	untext	Wooden (Hidden texture) GT im0 im1 all	disc	untext
EpicFlow [103]	55.0	0.12 74	0.36 88	0.09 60	0.25 62	0.85 70	0.21 72	0.39 54	1.00 63	0.25 66	0.19 55	1.01 64	0.11 51
ComplOF-FED-GPU [35]	56.1	0.11 54	0.29 58	0.10 78	0.21 30	0.78 58	0.14 18	0.32 42	0.79 46	0.17 15	0.19 55	0.99 63	0.11 51
Classic++ [32]	57.4	0.09 30	0.25 36	0.07 15	0.23 51	0.78 58	0.19 53	0.43 60	1.00 63	0.22 57	0.20 61	1.11 72	0.10 46
Steered-L1 [120]	57.5	0.09 30	0.22 16	0.08 42	**0.14** 1	0.49 2	0.12 5	0.28 24	0.69 32	0.16 12	0.18 50	1.06 67	0.09 25
HBM-GC [106]	57.9	0.14 89	0.28 51	0.12 91	0.26 68	0.69 37	0.22 75	0.34 45	0.75 39	0.22 57	0.21 67	0.77 28	0.15 75
Aniso. Huber-L1 [22]	58.5	0.10 44	0.28 51	0.08 42	0.31 81	0.88 75	0.28 86	0.56 79	1.13 72	0.29 78	0.20 61	0.92 58	0.13 65
TF+OM [100]	59.3	0.10 44	0.26 42	0.07 15	0.22 37	0.66 30	0.19 53	0.36 49	0.78 44	0.39 86	0.20 61	0.89 53	0.13 65
Average endpoint error	**avg. rank**	**Grove (Synthetic) GT im0 im1 all**	**disc**	**untext**	**Urban (Synthetic) GT im0 im1 all**	**disc**	**untext**	**Yosemite (Synthetic) GT im0 im1 all**	**disc**	**untext**	**Teddy (Stereo) GT im0 im1 all**	**disc**	**untext**
EpicFlow [103]	55.0	0.89 64	1.31 74	0.69 55	0.53 58	1.31 53	0.34 49	0.10 4	**0.11** 1	0.17 13	0.67 54	1.43 57	0.87 59
ComplOF-FED-GPU [35]	56.1	0.89 64	1.29 66	0.73 63	1.25 90	1.74 87	0.64 85	0.14 47	0.13 32	0.30 80	0.64 51	1.50 61	0.83 53
Classic++ [32]	57.4	0.87 57	1.30 68	0.66 53	0.47 42	1.62 76	0.33 43	0.17 81	0.14 54	0.32 90	0.79 75	1.64 71	0.92 65
Steered-L1 [120]	57.5	0.89 64	1.24 55	0.91 76	1.71 106	1.68 81	0.94 101	0.26 112	0.18 92	0.71 115	1.06 90	1.80 84	1.64 98
HBM-GC [106]	57.9	0.67 29	0.97 26	0.52 35	0.63 70	0.81 7	0.44 65	0.22 105	0.19 103	0.36 96	0.54 38	1.21 46	0.78 48
Aniso. Huber-L1 [22]	58.5	0.84 54	1.20 52	0.70 58	0.39 23	1.23 47	0.28 18	0.17 81	0.15 64	0.27 64	0.64 51	1.36 50	0.79 49
TF+OM [100]	59.3	0.98 82	1.31 74	1.03 85	0.56 62	1.55 73	0.33 43	0.16 72	0.17 82	0.27 64	0.76 67	1.59 69	0.98 72

Figure 2: A segment of the Middleburry ranking table for end-point error. The proposed method is denoted as 'Steered-L^1'.

	Average rank	Army	Mequon	Schefflera	Wooden	Grove	Urban	Yosemite	Teddy
LDOF	80.5	$0.12_{(74)}$	$0.23_{(84)}$	$0.43_{(60)}$	$0.45_{(98)}$	$1.01_{(86)}$	$1.10_{(86)}$	$0.12_{(27)}$	$0.94_{(86)}$
CLG-TV	69.5	$0.11_{(54)}$	$0.32_{(84)}$	$0.55_{(77)}$	$0.25_{(78)}$	$0.92_{(71)}$	$0.47_{(42)}$	$0.17_{(81)}$	$0.74_{(65)}$
Improved $TV-L^1$	63.8	$0.09_{(30)}$	$0.20_{(28)}$	$0.53_{(73)}$	$0.21_{(67)}$	$0.90_{(67)}$	$1.51_{(101)}$	$0.18_{(88)}$	$0.73_{(62)}$
Steered-L^1	57.5	$0.09_{(30)}$	$0.14_{(1)}$	$0.28_{(24)}$	$0.18_{(50)}$	$0.89_{(64)}$	$1.71_{(106)}$	$0.26_{(112)}$	$1.06_{(90)}$

Table 2: AEPE comparison for four algorithms including the algorithm proposed in this paper. Numbers in brackets indicate the ranking of the specific image sequence results, for example the results of the 'Mequon' sequence of our algorithm is ranked first. Numbers in blue indicate the highest rank.

AEPE for those methods at the time of writing this paper, the AEPE values are copied directly from the ranking table. Numbers between brackets indicate the ranking position for the particular image sequence.

It can be noticed that our algorithm performs better on non-synthetic image sequences. This can be seen clearly by comparing the AEPE and rank of individual non-synthetic image sequences like 'Mequon' (ranked 1st.) and 'Army' (ranked 30th.) in comparison with 'Urban'(ranked 106th.) and 'Yosemite'(ranked 112th.).

The estimated optical flow can be qualitatively assessed by visualising the displacement field. The colour code used in this paper is depicted in Figure-3 [23], where the direction of displacements is coded by the hue, and the magnitude of the displacements is coded by the saturation. Figure-4 depicts several examples for colour-coded results of the estimated optical flow field alongside their colour-coded ground truth for comparison.

5.2.2 *MPI-Sintel Dataset*

MPI-Sintel [30] is another dataset used to test optical flow methods. It is a synthetic image sequence taken from an animated 3D short film, it contains complex motion with varied textures. Images used in this

Figure 3: Colour code used to visualise the optical flow displacement fields.

dataset are rendered in three 'passes'. The first pass is 'albedo' which is the simplest rendering and does not contain illumination effect and has a piecewise constant colour. Hence, the data (brightness) constancy assumptions holds across the whole image. The second pass is the 'clean' pass which includes illumination effects (e.g. shading, specular reflections). The final pass is the one that matches the 'final' version of the film. This pass includes more complex effects and adds motion blur, atmospheric effect, colour correction, etc. This dataset is more challenging due to the inclusion of large motion and occlusion.

It was reported in [30] that methods with high-ranking on the Middlebury dataset have more difficulty estimating optical flow on this dataset. For example on the Middlebury ranking table the method 'Improved-TV-L^1' has an average rank of 63.8, which is much higher

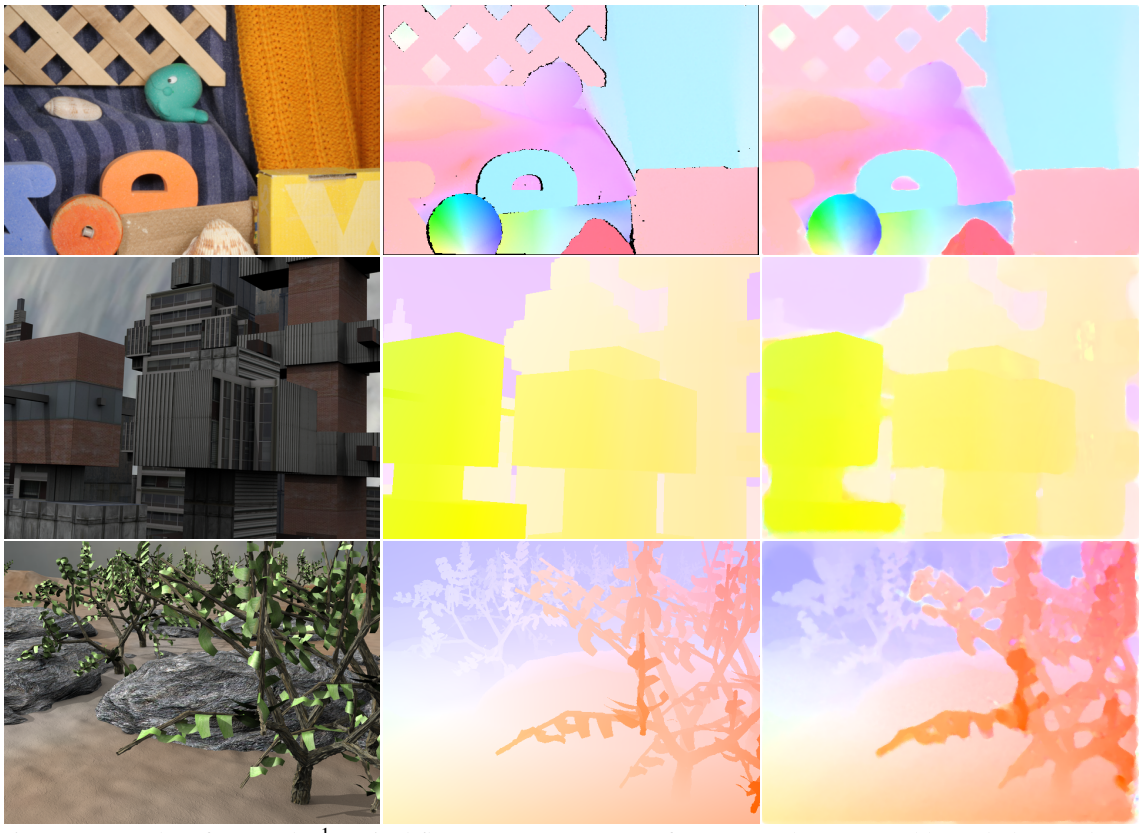


Figure 4: Results of Steered-L^1 optical flow. Image sequences from top to bottom 'RubberWhale', 'Urban3', 'Grove3'. Left column: First frame of the image sequence taken from the Middlebury dataset. Middle Column: Colour-coded ground truth. Right column: Colour-coded results.

than the ranking for LDOF [28] which has an average rank of 80.5. However, the same methods have an opposite order in the MPI-Sintel table, where LDOF is ranked higher than the Improved-TV-L^1. In this subsection we try to examine the effect of different levels of rendering on the estimation of optical flow using the proposed method in this paper. To this end the algorithm is applied to several image sequences at the clean and the final passes. The results of the estimation are compared both qualitatively and quantitatively. Table-3 illustrates the results of applying the algorithm for several examples from the training dataset of the MPI-Sintel benchmark.

Image clip	clean	final
alley_1	0.18	0.19
ambush_5	1.72	2.78
bamboo_1	0.23	0.23
Average	0.71	1.07

Table 3: AEPE for selected frames from MPI-Sintel dataset.

The AEPE results of our method now on the MPI-Sintel is 10.864 for the clean pass and 12.277 for the final pass. The relatively high AEPE in the case of this benchmark rate can be attributed to the complex motion and the effects included in this sequence, such as shading, specular reflection, motion blur, etc. The 'staircasing' effect, which is induced by the use of the L^1 norm, also contributes to the error rate. To deal with these issues and to improve the performance of this algorithm several suggestions are discussed in Section-6.

6 CONCLUSION AND FUTURE WORK

In this paper we have introduced a modified total variation L^1 norm to estimate optical flow denoted as 'Steered-L^1', which can be used to enhance the fill-in effect and hence the estimation accuracy in algorithms following the primal-dual formulation. In the proposed algorithm, the eigenvectors of the structure tensor are used to steer the displacement field derivatives into two components, one orthogonal and the other parallel to the local structure. This improves the fill-in performance of the total variation L^1 optical flow which reduces the error rate and improves the accuracy of estimation. It was shown experimentally that the utilisation of this steered norm improves the performance of the optical flow estimation and decreases the error in computation. Additionally, a high accuracy data term is used in the spirit of the delayed linearisation data constancy term proposed by Brox et al. [2]. This data term is augmented with image gradient to improve

Figure 5: Results on selected frames of MPI-Sintel. [30] Displacement field computed between the first frame (frame_0001) and the second (frame_0002). Top to bottom rows: alley_1, ambush_5, bamboo_1. Left column to right: First frame (frame_0001), ground truth, clean, final.

the robustness to illumination changes. Moreover an 'Extended Intermediate Filtering' EIF is proposed to enhance the displacement field estimation.

Despite the improved performance that this algorithm provides, it still suffers from a high AEPE in some scenarios. Further improvements are being investigated to render a more accurate optical flow estimation. In the next step of this research, other colour spaces are to be investigated such as the HSV (hue, saturation and value) colour space. The different channels of the HSV offer a more robust photometric performance [11]. This is expected to help deal with the illumination effects such as shadows and shading, and improve the performance especially in test benchmarks that include many such effects (e.g. MPI-Sintel).

The total variation L^1 is a piecewise constant function, hence it encourages a piecewise smooth solution for the displacement field. This produces artificial boundaries in the estimated optical flow field, a phenomena denoted as the 'stair-casing effect' [31]. In relatively smooth areas the performance of the quadratic norm is superior to the L^1 norm. To address this issue we are going to investigate the proposed method using the 'Huber-L^1' norm [31]. The Huber-L^1 norm behaves as the L^1 norm in areas with high gradients (i.e. motion boundaries), and behaves as a quadratic L^2 norm in the areas with lower gradients, hence offering enhanced performance [31], [32]. It will be interesting to investigate the performance of steering the Huber-L^1 norm using the structure tensor as was done with the L^1 in this paper.

Algorithms following the primal-dual formulation are generally used if speed of implementation is needed. These algorithms enable easy parallelisation on graphical hardware [9], [6], [7]. The current algorithm was implemented in MATLAB and the speed of implementation was not of concern. For example the time needed to estimate the optical flow displacement field for a colour image sequence of size 380×420 is around 457 seconds. A C/C++ and parallel implementation of this algorithm is going to be investigated in the future.

7 REFERENCES

[1] Horn, B. K. P. and Schunck, B. G. Determining optical flow. *Artificial Intelligence*, 17:185–203, 1981.

[2] Brox, T., Bruhn, A., Papenberg, N., and Weickert, J. High accuracy optical flow estimation based on a theory for warping, May 2004.

[3] Barron, J. L., Fleet, D. J., and Beauchemin, S. S. Performance of optical flow techniques. *InternationalL Journal of Computer Vision*, 12:43–77, 1994.

[4] Bruhn, A., Weickert, J., and Schnörr, C. Lucas/kanade meets horn/schunck: Combining local and global optic flow methods. *International Journal of Computer Vision*, 61:211–231, 2005.

[5] Chambolle, A. An algorithm for total variation minimization and applications. *J. Math. Imaging Vis.*, 20(1-2):89–97, January 2004.

[6] Zach, C., Pock, T., and Bischof, H. A duality based approach for realtime tv-l1 optical flow. In *Proceedings of the 29th DAGM conference on Pattern recognition*, pages 214–223, Berlin, Heidelberg, 2007. Springer-Verlag.

[7] Werlberger, M., Pock, T., and Bischof, H. Motion estimation with non-local total variation regularization. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, San Francisco, CA, USA, June 2010.

[8] Wedel, A., Pock, T., Zach, C., Bischof, H., and Cremers, D. *Statistical and Geometrical Approaches to Visual Motion Analysis: International Dagstuhl Seminar, Dagstuhl Castle, Germany, July 13-18, 2008. Revised Papers*, chapter An Improved Algorithm for TV-L 1 Optical Flow, pages 23–45. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[9] Drulea, M. and Nedevschi, S. Total variation regularization of local-global optical flow. In *Intelligent Transportation Systems (ITSC), 2011 14th*

International IEEE Conference on, pages 318–323, Oct 2011.

[10] Bruhn, A. and Weickert, J. Towards ultimate motion estimation: Combining highest accuracy with real-time performance. In *10th IEEE International Conference on Computer Vision (ICCV 2005), 17-20 October 2005, Beijing, China*, pages 749–755, 2005.

[11] Zimmer, H., Bruhn, A., and Weickert, J. Optic flow in harmony. *Int. Journal Comput. Vision*, 93(3):368–388, July 2011.

[12] Lucas, B. D. and Kanade, T. An iterative image registration technique with an application to stereo vision (ijcai). In *Proceedings of the 7th International Joint Conference on Artificial Intelligence (IJCAI '81)*, pages 674–679, April 1981.

[13] Papenberg, N., Bruhn, A., Brox, T., Didas, S., and Weickert, J. Highly accurate optic flow computation with theoretically justified warping. *Int. J. Comput. Vision*, 67(2):141–158, April 2006.

[14] Black, M. J. and Anandan P. A framework for the robust estimation of optical flow. In *IEEE ICCV*, pages 231–236, 1993.

[15] Perona, P. and Malik, J. Scale-space and edge detection using anisotropic diffusion. *IEEE Trans. Pattern Anal. Mach. Intell.*, 12(7):629–639, July 1990.

[16] Sun, D., Roth, S., Lewis, J. P., and Black, M. J. *Computer Vision – ECCV 2008: 10th European Conference on Computer Vision, Marseille, France, October 12-18, 2008, Proceedings, Part III*, chapter Learning Optical Flow, pages 83–97. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[17] Dosovitskiy, A., Fischer, P., Ilg, E., Häusser, P., Hazirbas, C., Golkov, V., Smagt, P. v.d., Cremers, D., and Brox, T. Flownet: Learning optical flow with convolutional networks. In *IEEE International Conference on Computer Vision (ICCV)*, Dec 2015.

[18] Chen, Z., Jin, H., Lin, Z., Cohen, S., and Wu, Y. Large displacement optical flow from nearest neighbor fields. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2013.

[19] Revaud, J., Weinzaepfel, P., Harchaoui, Z., and Schmid, C. EpicFlow: Edge-Preserving Interpolation of Correspondences for Optical Flow. In *Computer Vision and Pattern Recognition*, 2015.

[20] Weinzaepfel, P., Revaud, J., Harchaoui, Z., and Schmid, C. DeepFlow: Large displacement optical flow with deep matching. In *ICCV 2013 - IEEE Intenational Conference on Computer Vision*, pages 1385–1392, Sydney, Australia, December 2013. IEEE.

[21] Brox, T., Weickert, J., Burgeth, B., and Mrázek, P. Nonlinear structure tensors. *Image Vision Comput.*, 24(1):41–55, January 2006.

[22] Roth, S. and Black, M. J. Steerable random fields. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8, Oct 2007.

[23] Baker, S., Scharstein, D., Lewis, J. P., Roth, S., Black, M. J., and Szeliski, R. A database and evaluation methodology for optical flow. *Int. J. Comput. Vision*, 92(1):1–31, March 2011.

[24] Freeman, W. T. and Adelson, E. H. The design and use of steerable filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13:891–906, 1991.

[25] Mileva, Y., Bruhn, A., and Weickert, J. Illumination-robust variational optical flow with photometric invariants. In *In DAGM-Symposium, LNCS 4713*, pages 152–162, 2007.

[26] Andrews, R. J. and Lovell, B. C. Color optical flow. In *Proceedings Workshop on Digital Image Computing*, pages 135–139, 2003.

[27] Durand, F. and Dorsey, J. Fast bilateral filtering for the display of high-dynamic-range images. In *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '02, pages 257–266, New York, NY, USA, 2002. ACM.

[28] Brox, T. and Malik, J. Large displacement optical flow: Descriptor matching in variational motion estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(3):500–513, 2011.

[29] Scharr, H., Black, M.J., and Haussecker, H.W. Image statistics and anisotropic diffusion. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 840–847 vol.2, Oct 2003.

[30] Butler, D.J., Wulff, J., Stanley, G.B., and Black, M.J. A Naturalistic Open Source Movie for Optical Flow Evaluation. In Andrew Fitzgibbon, Svetlana Lazebnik, Pietro Perona, Yoichi Sato, and Cordelia Schmid, editors, *ECCV*, volume 7577, pages 611–625, 2012.

[31] Werlberger, M., Trobin, W., Pock, T., Wedel, A., Cremers, D., and Bischof, H. Anisotropic Huber-L1 optical flow. In *Proceedings of the British Machine Vision Conference (BMVC)*, London, UK, September 2009.

[32] Black, M. J., Sapiro, G., Marimont, D. H., and Heeger, D. Robust anisotropic diffusion. *Trans. Img. Proc.*, 7(3):421–432, March 1998.

ISSN 2464-4617 (print) WSCG 2016 - 24th Conference on Computer Graphics, Visualization and Computer Vision 2016
ISSN 2464-4625 (CD-ROM)

Cytological Low-Quality Image Segmentation Using Nonlinear Regression, K-means and Watershed

Ramon A. S. Franco, Paulo S. Martins, Marco A.G. de Carvalho
University of Campinas (UNICAMP)
Brazil , 13484-332, Limeira, SP
{ramon,paulo,magic}@ft.unicamp.br

ABSTRACT

Since 1950, conventional cytology uses glass slides for microscopic analysis of cervical cells, in order to perform Pap Test. Such method yields low-quality images and overlapping cells, which both hampers their analysis and classification. Several countries use a modern method for the realization of Pap test called ThinPrep because it offers high- quality images and overcomes the problem of overlapping cells. ThinPrep facilitated the development of advanced image processing techniques for segmentation and classification of cervical cells. However, this method is not used by most of the developing countries of the world due to its relative high cost. This paper presents an algorithm for segmenting digital images obtained from conventional cytology method on glass slides. The technique uses Watershed Transform and K-Means Clustering in order to find cell markers or seeds. Nonlinear regression is applied as a way to refine the markers and to allow again the Watershed Transform utilization. We apply the technique in 10 glass slides of pap smears with a total of 67 cells. Our proposed technique has a promising performance in terms of accuracy of about 85%.

Keywords

Pap test, Image analysis, Watershed, K-means

1 INTRODUCTION

Cervical cancer remains the second leading cancer affecting women in developing countries [Glo01a]. The Pap Test, also known as oncological cytology or exfoliative cytology, has been used for the collection of human papillomavirus in the battle (i.e. prevention and diagnosis) against cervical cancer since 1950.

The Pap Test is a method developed by the physician George Papanicolaou for identifying neoplastic malignant or pre-malignant cells that precede the development of cancer [Mor01a]. This technique was originally developed to prevent cervical cancer. The cells are harvested in the region of the external orifice and endocervical canal, placed in a transparent glass slide, stained and taken for examination under the microscope.

Trained personnel can distinguish normal cells from malignant cells, such as those with indications of precancerous lesions [Nai01a]. However, the conventional Pap test technique has its limitations.

One of its limitations is the occurrence of false negatives (FN), i.e. when abnormal cells present in the test remain undetected. Aspects such as the manual spreading of cells leading to cell fragmentation on the glass slides is one of the reasons why false negatives occur in the Pap test. This normally occurs after obtaining cervical cells, where the physician must transfer cells using the spatula against the surface of the glass slides. The remaining cell assembly is disposed on the lamina, possibly leading to overlapped cells. If an abnormal cell is hidden under a healthy one, the pathologist would have difficulty locating them. Fig. 1 shows the appearance of a layer of cells collected by this method [Bud01a].

As we can see in Fig. 1, the cellular material is spread over the entire area, resulting from the friction of the wooden spatula on the laminated surface. To overcome the problem of overlapping cells of the cervix, a more efficient technology called ThinPrep was developed.

ThinPrep is a technology that mechanically separates the cervical cells by centrifugation, providing visibility of the cells and avoiding overlapping of cells [Loz01a]. In order to use the ThinPrep technology, it is necessary to change the conventional collection of glass by another collection called LCB [Ans01a]. Despite the advantages of the ThinPrep technology, it is not used by most countries due to its prohibitive costs and the lack of infrastructure needed for its implementation [Ama01a].

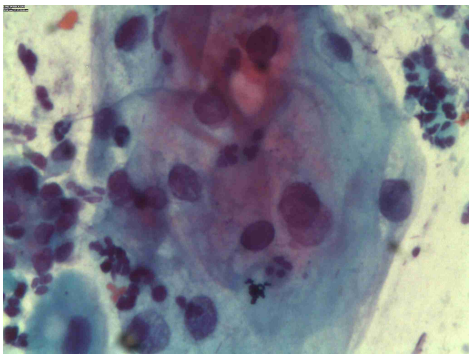


Figure 1: Glass slide of conventional Pap test

Much of recent research on image segmentation and image classification is founded on the analysis of images captured by the ThinPrep method, which considerably facilitates the task of pre-processing.

The goal of this paper is to segment images of cervical cells of low quality, obtained from conventional cytology. We apply clustering techniques and Watershed. A proposed solution is presented to improve the quality of segmentation. The proposed method combines K-means and Watershed in order to identify cervical cells, and it uses nonlinear regression for preprocessing of the histogram.

This paper is organized as follows: Section 2 presents a brief review of early work. The approach adopted is presented in Section 3. Section 4 shows initial results and discussions. Finally, in Section 5, we present the main conclusions and future work.

2 RELATED WORK

A nucleus and cytoplasm contour detector (NCC Detector) is presented by Pai et al. to automatically detect the cytoplasm and nucleus contours of a cell in a cervical smear image [Pai01a]. This detector has two different phases, according to the object to be segmented. In the cytoplasm detector, ATD method (a thresholding method) is used in order to draw the cytoplasm contour. The nucleus detector phase is composed of three stages: gradient calculation, the maximal gray-level-gradient-difference (MGLGD) method (proposed to sever the nucleus from the cytoplasm) and the contour connection. The experiments were accomplished with 50 cervical smear gray-level images, of 128x128 pixels. The results show that the NCC Detector is superior to two existing methods, the gradient vector flow-active contour model [ChaV01a] and the edge enhancement nucleus and cytoplasm contour detector.

Yung-Fu et al. proposed a method based on five steps well defined: image acquisition and categorization, image editing and processing, morphometry (contour segmentation of cell nucleus and cytoplasm, measurement and analysis), Support Vector Machine (SVM) classification and assessment of diagnostic performance[Yun01a]. The main goal is classifying four different types of cells and to discriminate dysplastic from normal cells. Besides, two experiments were conducted to verify the classification performance and results showed that average accuracies about 97%. The authors used a set of performance metrics and presented a good number of tables that shows the classification and diagnostic performance.

A method for automatic cervical cancer cell segmentation and classification was proposed by Chankong et al. [Cha01a]. His method provides a cell segmentation that separates nucleus, cytoplasm and background. The approach uses Fuzzy C-means (FCM) clustering technique and the results achieved a segmentation accuracy around 95%, according to the chosen classifier. The experiments have not included full-glass slides segmentation, i.e., the authors work with selected regions. The segmentation and classification performances were compared with C-means clustering and Watershed techniques. The comparison analysis showed that the proposed approach results in good performance and is better than the cited work.

Ushizima et al. [Ush01a] developed algorithms for quantitative analysis and pattern recognition from 2D cervical cells images. They proposed a pre-processing step, based on adaptive histogram equalization and mean shift technique, in order to make homogeneous the image regions and have an image with good contrast. The overall cytoplasm segmentation is accomplished using Watershed Transform. Before that, it is implemented some tasks in order to find the Watershed seeds, including clustering and hole filing. The authors provide a set of automated tools capable of detecting multiple cells obtained from ThinPrep Pap Test, including ROI (region of interest) selection, noise minimization and cell classification. A difficulty of the proposed method consists on the cytoplasm segmentation, according to the paper, generating a low accuracy.

In essence, our work differs from previous work by the one or more of the following features:

- *Low-quality images*: The set of images adopted in this work, as mentioned before, were obtained directly from relatively low-resolution cameras and standard microscopes, which is the reality of many developing countries; this contrast with much of the work which uses ThinPrep as the basis for the segmentation procedure;

- *Conventional testing*: The procedure employed is the conventional testing which uses manipulation of cells on a glass slide. The challenge with this process is the resulting undesired elements in the glass

slide, such as cell fragments, which may negatively interfere with the segmentation process;

- *Watershed transform*: We used the Watershed transform, which is arguably a satisfactory method for finding non-uniform contours such as the ones from cells. Clearly, there are various methods that can be used to segment images, or even a combination of methods. Ongoing work is experimenting with such variations; Whereas there has been a relative large number of work employing the Watershed, we are not aware of other projects dealing with cytological segmentation that uses Watershed twice or in a chained fashion, whereby the output of the first is the input to the second (Section 3);

- *Nonlinear regression:* Nonlinear regression, despite its relative simplicity and low computational overheads, proved to be an effective method for clearing the background image, and thus providing a much easier path for the ensuing segmentation process. Most work applies Otsu [Liu01a] in this stage, which we applied in an earlier stage of this work. However, the results obtained with the nonlinear regression filter were far superior, and therefore it was the method of our choice for this work.

Furthermore, it is not the focus of this work (for now) to experiment with classification of cells; our short term goals are to increase and maximize the accuracy of the method by exploring new variants to the proposed approach.

3 PROPOSED APPROACH

In this section we describe the approach employed to improve the quality of segmentation in low-quality images. It consists of 11 steps, which are illustrated in Fig. 2 and described through the remainder of this section.

To facilitate the discussion, these steps are grouped in three major classes, i.e. 1) Acquisition and categorization, 2) Image processing and 3) Cell Segmentation.

Acquisition and Categorization

The nucleous and cell morphology was used to calculate the ground truth in each image. The ratio of the cell area NC_r is given by:

$$NC_r = \frac{N_a}{N_a + N_c} \tag{1}$$

where N_a is the nucleus area and N_c is the cytoplasm area.

Image Preprocessing

Considering that images were of poor resolution, with non-homogeneous contrast and brightness, it was necessary to perform filtering on the image. Three filters were applied:

- *Bilateral filtering:* We applied the following nonlinear bilateral filter to preserve image energy and the image contours while simultaneously reducing noise:

$$I_f(x) = \frac{1}{W_p} \sum_{x_i \varepsilon \Omega} I(x_i) f_r(||I(x_i) - I(x)||) g_s(||x_i - x||) \tag{2}$$

  where: I_f is the filtered image; I is the original input image to be filtered; x are the coordinates of the current pixel being filtered; Ω is the window centered in x; f_r is the range kernel for smoothing differences in intensities; g_s is the spatial kernel for smoothing differences in coordinates. The range parameter σ_r was set to 9. The spatial parameter σ_d was set to 75.

- *Median filtering:* The median filter was applied with a kernel 3x3, in order to remove noise while preserving the edges and other details;

- *Unweighted average:* All images show a significant background representing the glass slide. The cytoplasm scattered across the glass slide and the undesired background in Fig. 3 would hinder the identification of tonal groups by the clustering algorithm, if not removed.

  Therefore, we decided to apply a nonlinear moving average filter in order to remove the background of the image (i.e. glass slide). We applied the equation MA = P_m + P_{m-1} +.. .+ $P_{m(n-1)}$)/ N where P_m is the average amount N and the number of samples [YaL01a].

  This nonlinear filter smoothens the histogram creating a harmonic function and enabling the determination of the maximum peaks. Consequently, the background image is removed as illustrated in Fig. 4. Thus, it was possible to automatically establish the threshold of each one of the glass slides image. Fig. 5 illustrates the image histogram before we applied the unweighted average filter, and Fig. 6 the histogram after the filtering.

  Clearly, the peak on the right side of the graph mainly belongs to a set of data pertaining to the bottom of the glass slide. These pixels were then removed before starting the image processing step.

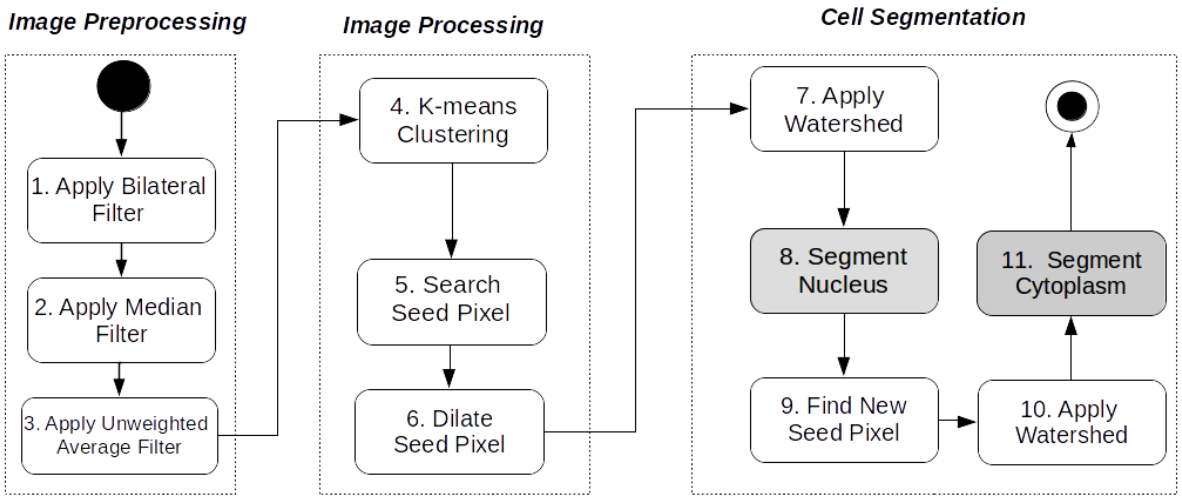


Figure 2: Activity Diagram of the Segmentation Process

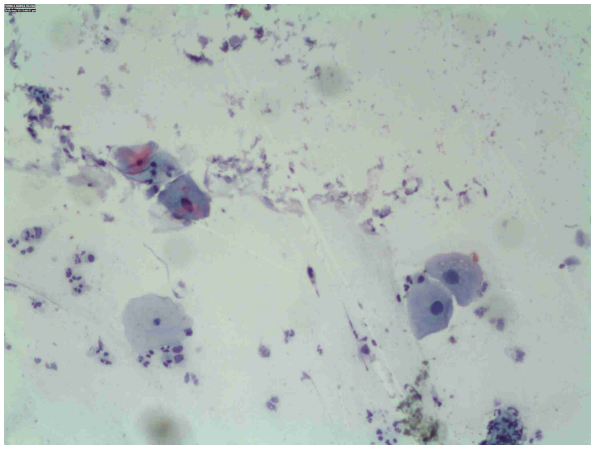


Figure 3: Low-quality image of cervical cells in glass slide

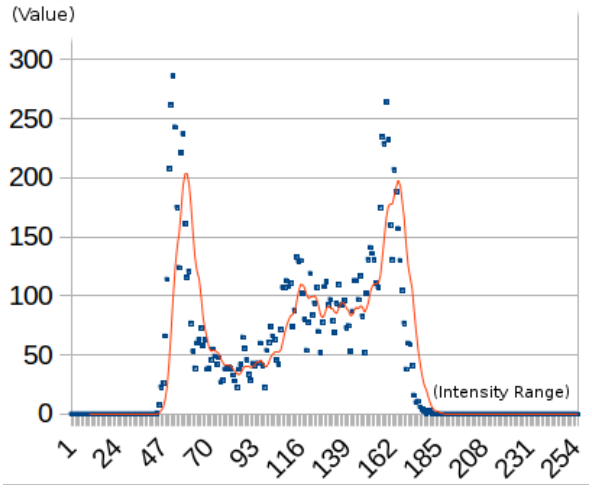


Figure 5: Original histogram (pixel frequency vs gray-level value)

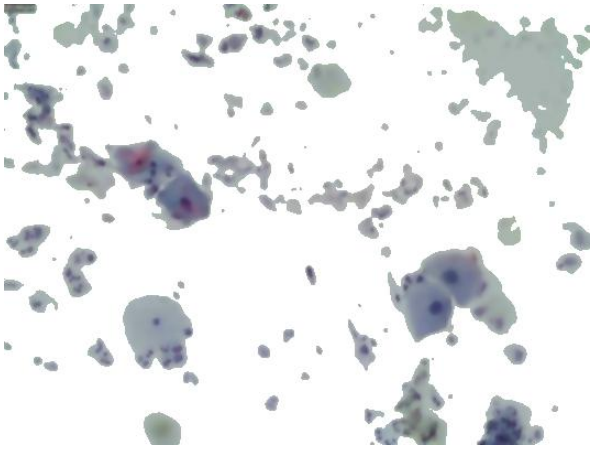


Figure 4: Isolating the cells from the glass slide

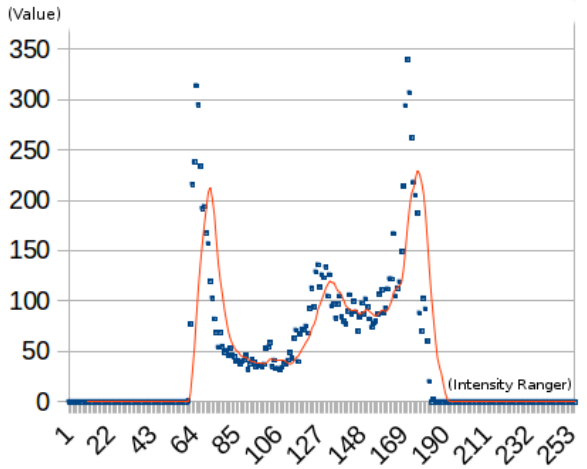


Figure 6: Histogram modified by the unweighted moving average filter

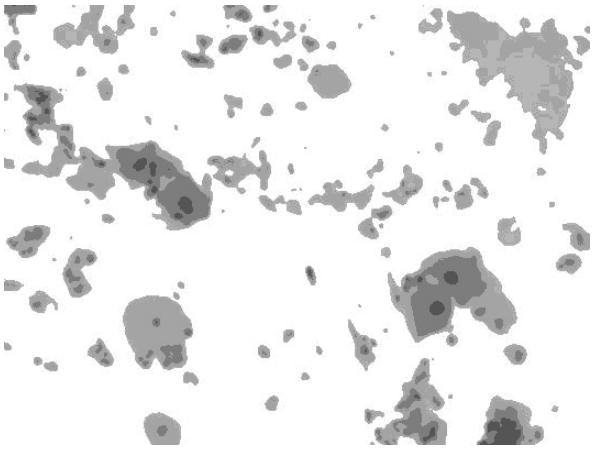


Figure 7: The image after clustering

Image Processing

After preprocessing the images, the K-Means clustering algorithm is applied in order to create distinct gray-scale groups [Har01a]. The goal of using K-Means is exclusively to find distinct gray-scale levels of seed pixels for the segmentation using the Watershed algorithm [Beu01a]. The number of groups selected was seven, as this number had shown the best segmentation results in a number of experiments varying the number of groups. Fig. 7 shows the results observed after image clustering.

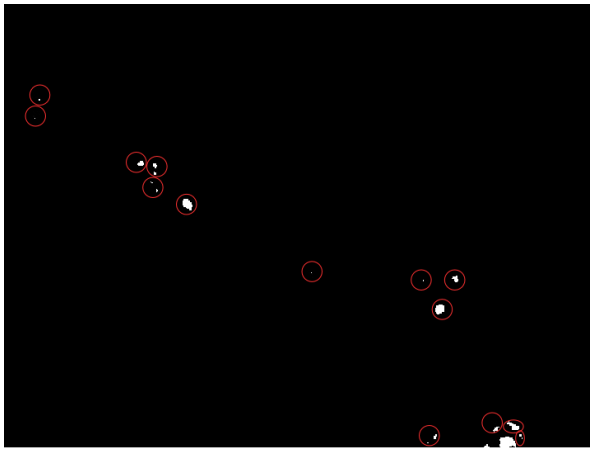


Figure 8: Pixels seed

After clustering, the cytoplasm still does not show well since it is illustrated within four different shades of gray (i.e. groups of gray levels). Note that the background is one among the four groups. Note also that this technique also detects other elements such as cell fragments, resulting from the mechanical handling of the samples on the glass slide. On the other hand, the nuclei has successfully been isolated in the image, i.e. within the darkest areas. Thus, a search algorithm was used to determine the two most prominent (darkest) groups, as such groups are the ones that capture the cell nuclei.

The pixels at the coordinates output by the clustering process were then dilated with a scale factor 3 to facilitate their identification as seed pixels, and thereby the most significant nuclei. These pixels were used in the following step in the Watershed algorithm, to help determination of the masks and the unknown area of the image. Fig. 8 shows some of the resulting pixels seeds.

Cell Segmentation

Cell segmentation was carried out in two steps, each consisting of one application of the Watershed. This process ensured the proper segmentation o the cytoplasm:

- *Watershed 1*: The goal of this first Watershed step was to identify the nuclei boundaries. The seed pixels were the ones found earlier by the clustering process. A good way to understand the Watershed is by comparing its operation to that of a flooding of a water basin. In the Watershed method, an image is segmented by constructing the catchment basins, or lakes, of the image. The image is flooded starting from the seeds, where each seed correspond to a lake, until the whole image has been flooded. A dam is built between lakes that meet with others lakes. At the end of flooding process, we obtain one region for each catchment basin of the image [Beu01a].
- *Watershed 2*: In a second step, subsequently, we reapplied the Watershed algorithm. However, this time using as the new seed pixels the nuclei detected above. In this case the goal is to find the cytoplasm boundaries.

The K-Means Clustering algorithm was able to find the seed pixels that were subsequently used by the first implementation of the Watershed Transform (Watershed 1, as cited above

4 RESULTS

In this section we show a comparison analysis of the results obtained by manual segmentation carried out by the domain expert (i.e. pathologist) against the segmentation obtained by the automatic segmentation approach. In essence, the area of the cytoplasm and the nucleous found by the pathologist were compared to the respective areas found by the automatic segmentation. This procedure was repeated throughout 10 glass slides.

The details of this analysis are shown in Table 1, where A_{cp} is the average area of cytoplasm identified by the pathologist (manual segmentation); A_{np} is the average area of nuclei identified by the pathologist (manual segmentation); A_{cs} is the average area of cytoplasm identified by the segmentation algorithm (automatic segmentation), and A_{ns} is the average area of nuclei identified

Glass Slide #	# Cells	A_{cp}	A_{np}	A_{cs}	A_{ns}	Cytoplasm error (%)	Nuclei error (%)	Accuracy (%)
1	6	10551.63	558.88	11110.51	730.51	5.03	23.50	81.53
2	5	63934	5285.99	58798.46	5536.57	8.73	4.53	95.79
3	14	16823.85	1041.67	17328.23	1431.19	2.91	27.22	75.69
4	1	1237.81	76.74	1348.14	137.5	8.18	44.19	63.99
5	7	18061.66	1118.41	18676.37	1568.69	11.71	19.52	68.67
6	5	19299.47	1195.15	20024.51	1706.19	6.12	24.08	82.04
7	6	4485.46	459.83	4448.96	699.65	0.81	34.28	66.54
8	8	6157.05	784.5	4788.19	748.96	22.23	0.74	78.51
9	6	29883.9	868.93	29816.27	742.56	0.23	14.54	76.15
10	9	6262.5	1086.11	6834.34	1184.5	8.37	8.31	74.09
Results	67	176697.33	12476.21	273174	14486.32	7.43	20.09	85.00

Table 1: Accuracy of the segmentation (Acp and Anp, manual segmentations; Acs and Ans, automatic segmentations)

by the segmentation algorithm (automatic segmentation). For example, in glass slide 7 six cells were identified; the total area of cytoplasm and the total area of nuclei are shown for both manual and computational procedures; the percentage cytoplasm error is 0.81, and the corresponding percentage nuclei error is 34.28; the accuracy for this slide is 66.5%. Considering all glass slides, the total accuracy was 85 %.

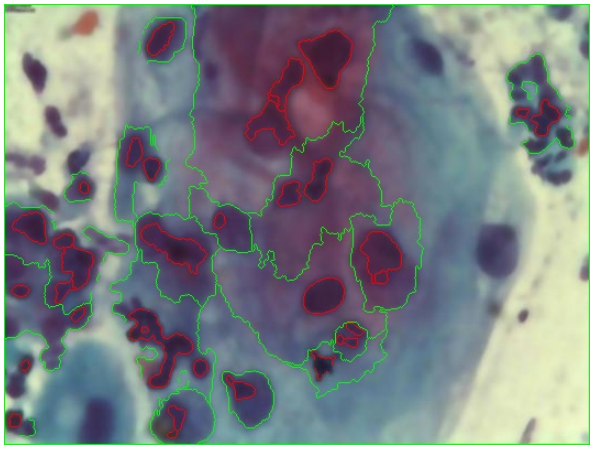


Figure 9: Final segmentation (glass slide 3)

As shown in Table 1, the segmentation accuracy was 85%. The method segmented 60% of the cells marked by the pathologist. On the other hand, it was able to segment 20% of the cells that were not identified by the pathologist.

Fig. 9 illustrates the final segmentation of the cytological low-quality image (glass slide 3) using nonlinear regression, K-Means clustering and Watershed Transform. As we can see, the algorithm segmented 15 cytoplasms and 28 nuclei, which were scattered throughout the image. 30% of the nuclei identified through automatic, image-processing segmentation, were not easily identifiable by pathologists because of the poor quality of the images.

5 CONCLUSION

This work is intended to improve the availability of current image processing technologies in countries and areas that are not able to afford modern collection methods such as Thin Prep.

The use of glass slide for conventional Pap test cytologies is quite common in developing countries, due to their low costs and easy implementation. However, the analysis of generated samples by image processing algorithms is a challenge: many of the images collected on glass slide are not exploited on account of their low quality and visibility. Without a method that is conceived to address such concerns, pathologists are not able to perform diagnostics using these images. This leads to extra losses, since the samples cannot be processed and have to be simply eliminated.

We performed image segmentation using images of poor quality, by means of nonlinear regression, K- means clustering and Watershed transform. We conducted experiments with 10 glass slides containing 67 cells previously measured by the pathologists. In addition, we achieved cell nuclei segmentation, which were not labeled by the pathologist by low visibility.

The segmentation accuracy was 85%. Considering that such images are currently discarded by the pathologist because of their low quality, this may already be deemed an acceptable result. However, in future work, we intend to carry out further experiments using complementary techniques to improve the accuracy as well as the overall citoplama's segmentation.

6 ACKNOWLEDGEMENTS

The authors would like to thank Synergy Telemedicine Company S.A. from Colombia for allowing the use of Pap smear slide images for investigation purposes. Ramon Franco is supported by Brazilian Agency CAPES - (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior).

7 REFERENCES

[Mor01a] Morrell, S., Taylor, R. and Wain, G. A study of Pap test history and histologically determined cervical cancer in NSW women, 1997-2003., J. Medical Screen., vol. 12, no. 4, pp. 190-196, 2005.

[Nai01a] Naib, Z. M. Pap Test, in Clinical Methods: The History, Physical, and Laboratory Examinations, 1990.

[Bud01a] Budge, M., Halford, J., Haran, M., Mein, J. and Wright, G. Comparison of a self-administered tampon ThinPrep test with conventional pap smears for cervical cytology, Aust. New Zeal. J. Obstet. Gynaecol., vol. 45, no. 3, pp. 215-219, 2005.

[Loz01a] Lozano, M., De Miguel, C. and Cousillas, Nuevas tecnologias en Citopatologia, in Sociedad Espanola de Anatomia Patologica, p.143, 2011.

[Ans01a] Anschau, F., Goncalves, M. A. G. Citologia Cervical em Meio Liquido Versus Citologia Convencional, Femina, vol. 34, no. 05, pp. 329-335, 2006.

[Ama01a] Amaya, M., Acosta, P., Mora, M. Citología convencional y en base liquida en muestra compartida de tomas cervocouterinas, Reperto Medico y Cirugia., vol. 24, no. April, pp. 41-47, 2016.

[Glo01a] GLOBOCAN. Estimate Cancer Incidence, Mortality and prevalence Wordwde in 2012, 2012. [Online]. Available: http://globocan.iarc.fr/Default.aspx.

[Liu01a] Liu, D. and Yu, J. Otsu method and K-means, in Proceedings - 2009 9th International Conference on Hybrid Intelligent Systems, HIS 2009, 2009, vol. 1, pp. 344-349.

[Pai01a] Pai, C. Chang, and Y. K. Chan, Nucleus and cytoplast contour detector from a cervical smear image, Expert Syst. Appl., vol. 39, no. 1, pp. 154-161, 2012.

[Yun01a] Yung-Fu, C. Semi-Automatic Segmentation and Classification of Pap Smear Cells, IEEE Journal of Biomedical and Health Informatics, vol. 18, no. 1, 2014.

[Cha01a] Chankong, T., Theera-Umpon, N. and Auephanwiriyakul, S. Automatic cervical cell segmentation and classification in Pap smears, Computer Methods and Programs in Biomedicine, vol. 113, no. 2, pp. 539-556, 2014.

[Ush01a] Ushizima, D. M., Gomes, A. H., Carneiro, C. M. and Bianchi, A. G. C. Automated Pap Smear Cell Analysis: Optimizing the Cervix Cytological Examination, in 2013 12th International Conference on Machine Learning and Applications, pp. 441-444, 2013.

[Yal01a] Ya-Lun. Chou, Statistical Analysis With Business and Economic Applications, 2nd ed. Mishawaka, 1975.

[Har01a] Hartigan, J. A. and M. A. Wong, A K-Means Clustering Algorithm, Appl. Stat., vol. 28, no. 1, pp. 100-108, 1979.

[Beu01a] Beucher, S. The Watershed Transformation Applied to Image Segmentation, in Proceedings of the 10th Pfefferkorn Conference on Signal and Image Processing in Microscopy and Microanalysis, 1992, pp. 299-314.

[ChaV01a] Chan, T. F., and Vese, L. A. Active contours without edges. IEEE Transactions on Image Processing, 10(2), 266-277, 2001.

[LiY01a] Liu, D. and Yu, J. Otsu method and K-means,Ó in Proceedings of the 9th International Conference on Hybrid Intelligent Systems, vol. 1, pp. 344-349, 2009.

ISSN 2464-4617 (print) WSCG 2016 - 24th Conference on Computer Graphics, Visualization and Computer Vision 2016
ISSN 2464-4625 (CD-ROM)

CellPathway: a Simulation Tool for Illustrative Visualization of Biochemical Networks

Matthias Reisacher
TU Wien, Austria
1125358@student.tuwien.ac.at

Ivan Viola
TU Wien, Austria
viola@cg.tuwien.ac.at

Mathieu Le Muzic
TU Wien, Austria
mathieu@cg.tuwien.ac.at

ABSTRACT

The molecular knowledge about complex biochemical reaction networks in biotechnology is crucial and has received a lot of attention lately. As a consequence, multiple visualization programs have been already developed to illustrate the anatomy of a cell. However, since a real cell performs millions of reactions every second to sustain live, it is necessary to move from anatomical to physiological illustrations to communicate knowledge about the behavior of a cell more accurately. In this publication we propose a reaction system including a collision detection algorithm, which is able to work at the level of single atoms, to enable simulation of molecular interactions. To visually explain molecular activities during the simulation process, a real-time glow effect in combination with a clipping object have been implemented. Since intracellular processes are performed with a set of chemical transformations, a hierarchical structure is used to illustrate the impact of one reaction on the entire simulation. The CellPathway system integrates acceleration techniques to render large datasets containing millions of atoms in real-time, while the reaction system is processed directly on the GPU to enable simulation with more than 1000 molecules. Furthermore, a graphical user interface has been implemented to allow the user to control parameters during simulation interactively.

Keywords: Molecular simulation, visualization system, collision detection, particle-based data, large data

1 INTRODUCTION

The usage of illustrative tools is an established approach to communicate knowledge of complex biochemical processes in cells to a broad audience. In the beginning, illustration artists had to create time consuming handmade animations combined with sophisticated visualization techniques to tell a structured story. The next step was to use software tools to create images showing complex molecular structures. While at the beginning, render processes took hours or days to complete, with increasing processing power it was possible to explore large scenes containing millions of atoms in real-time. However, millions of chemical reactions are performed every second in real cells to allow intercellular communication and to sustain living organisms. To communicate knowledge about complex intracellular processes which keep the cell alive, it is necessary to move from anatomical to physiological illustrations. Therefore, the next logical step is to use molecular reaction systems to simulate large scale reaction networks which are describing the physiology of a cell.

While this area has received a lot of attention lately, many tools to simulate and visualize molecules and reactions inside of a cell have been proposed in the last few years. Lately, particle-based simulators got more popular to imitate a realistic behavior of the molecules. Their general approach is to postpone most of the calculations and operations from the central processing unit to the graphics card through a GPU first approach by, for example, enabling GPU-to-GPU data flow. This is possible due to the modern, freely programmable GPUs. General-purpose GPU programming accelerates the performance of those systems immensely. That enables the simulation and visualization of large-scale scenes containing billions of atoms on an average computer. However, most of those approaches do not take global collision detection into account. This improves performance but leads to visible artifacts during the animation.

My goal is to extend a modern particle-based illustration tool with a basic molecular simulator and a collision detection system. Additionally, a visualization system to improve the user's awareness of biochemical processes and to display reaction networks inside of a specified area is implemented. Besides, the user should be able to interact with the system to optimize the learning effect. To implement those goals, more calculations per frame have to be executed, which has a significant impact on the performance. Especially the collision detection needs multiple processing steps. For load balancing, the simulation is only executed in a specific part of the scene, whereby the user can determine the size and the position of this area. Therefore, the program's requirements can be scaled down manually by the user to enable the simulation also on weaker computers.

WSCG 2016 conference proceedings
WSCG'2016, May 30 – June 3, 2016
Plzen, Czech Republic.

The system is based on the technique proposed by Le Muzic et al. [MAPV15]. While almost every aspect of the core visualization techniques are inherited, a simpler reaction system has been implemented. The quantitative simulation itself is calculated by the COPASI [HSG+06] API and the reaction system is working with an omniscient intelligence while using passive agents for dynamic simulation given by Kubera et al. [KMP10]. To enable a fast and easy change of the simulation system, the user is provided with a simple UI, whose implementation is inspired by the publication of Daniel Gehrer [D14].

Current techniques in mesoscale visualization of biochemical processes are including collision detection only partially or they are ignoring it at all for the sake of performance. Tools like ZigCell3D [CKMK13] or MegaMol [GKMRE14] are great for visualization but because the molecular participants do not collide and therefor don't interact with each other beside during a reaction, they are not able to showcase realistic animation of molecular crowding. Furthermore, visible artifacts occur. The main contribution of this work is to implement a three dimensional collision detection system which is able to detect the intersections of two or more objects at the level of single atoms. Additionally, an illustration technique using two adjustable cone-cut-objects in combination with a real-time glow effect is implemented to make complex processes visible even in dense scenes but without losing the impression of depth. Further, a hierarchical structure is used to illustrate intracellular process, by showing the impact one reaction has on the entire simulation. Since this project is based on the work proposed by Le Muzic et al., it also uses the Unity3D [WSUnity] engine. Unity is a cross-platform game engine which is also available for free in a limited, but still functional, scope. The provided user interface is also implemented in Unity. Simulated molecules can be downloaded from the public PDB database [WSPdb] and afterwards imported through the user interface.

2 RELATED WORK

We structure the prior work review in two parts. Firstly, we refer to agent-based simulation systems using a game-based environment. In the second part we relate with research techniques that employ with Monte Carlo-based simulation systems.

2.1 Agent-based Simulations in Game-like Environments

Using game-like environments to reduces the software development work-load is getting more and more popular among the visualization community [MAPV15]. A recent work on visualizing the anatomy of a cell in a multiscale approach has been proposed by Muzic et al. [MAPV15]. cellView has been implemented in the Unity3D [WSUnity] game engine and uses macromolecular datasets, which are modeled with the cellPACK [JAAGS15] tool. cellPACK is publicly available and enables the generation of large biomolecular structures. By using advanced GPU programming and acceleration techniques, such as hierarchical Z-buffer occlusion culling and a twofold level-of-detail approach, it is possible to render scenes containing billions of atoms. But since no molecular dynamics are supported, cellView can not be used to illustrate the physiology of a cell.

A tool to visualize agent-based simulations has been proposed by de Heras Ciechomski et al. [CKMK13]. An accurate simulation system is implemented in an interactive and game-like 3D environment to illustrate complex chemical processes at various zoom levels. Cellular reactions are modeled with a GUI and are represented as an SBGN [WSSbgn] network diagram. However, the rendering module does not use GPU programming and therefore, no real-time processing of large scenes is possible.

To enable real-time rendering of billions of molecules, the necessary calculations have to be done in parallel using advanced GPU programming. Such visual explanation tools have been proposed by Le Muzic et al. [MPSV14] and with CellUnity[D14]. A particle-based simulation system is used in combination with passive agents and an omniscient intelligence to simulated biochemical reactions in a story telling manner. Nevertheless, only a limited collision detection system has been implemented and the tool is not publicly available. Another simulation system using this approach is CellUnity [D14] It has been implemented in Unity3D and uses its physics engine to apply a fully collision detection during simulation. However, since the project is implemented entirely on the CPU, only small scenes containing a few hundred molecules can be used for simulation.

2.2 Monte Carlo based Simulation Systems

By using specialized Monte Carlo algorithms, MCell [SB01] is a popular tool to simulate chemical reactions in multiple compartments. The visualization tool CellBlender [WSCB], which is an addon for the open-source 3D computer graphics software Blender [WSB], enables a simple and fast way to model and edit the molecule designs of a simulation. Although, the simulation settings can be changed directly in Blender, no interactive storytelling approach can be used to present the outcomes.

Illustrative timelapse is a cross-platform simulation tool with the focus on multi-scale temporal illustrative visualization techniques. MCell is used to model and simulate biochemical processes while the visualization

part is implemented in Unity3D using the technique proposed by Le Muzic et al. [MPSV14] A combination of interactive temporal zooming and visual abstraction is used to communicate reaction processes in a storytelling manner. However, no collision detection has been implemented.

3 SIMULATION

In this section we describe the reaction system and the simulation algorithm, as well as implemented techniques for load balancing.

3.1 Global Simulation

Our molecular reaction system is based on the technique proposed by Le Muzic et al. [MPSV14]. While the individual molecules who are participating in the simulation process are implemented as passive agents, an omniscient intelligence (OI) is used to control molecular interactions. Passive agents are unable to start reactions autonomously, instead they can only receive reaction orders from an OI, which is tightly coupled with the quantitative simulation [MPSV14]. The system uses the COPASI API [HSG+06] as simulation engine, which is responsible to initiate new reactions in regular time intervals, whereby the interval length is specified by the user. Since each reaction has to be processed separately on the GPU, the number of created reactions per frame is limited to 20 to increase the performance. For each initiated reaction the reaction system searches for appropriate reactants, which are not already included in an open reaction. While the first reactant is picked randomly, the other molecules of the specific reaction are selected by their distance to the first molecule. Only the molecules closest to the first reactant are assigned to the reaction. Additionally, every reaction type can be linked to protein by the user. Thus, the reactants need to enter a random protein to perform the specific reaction.

To create the impression of chaotic behavior, the molecular movement during a reaction is created by interpolating direct motion with Brownian motion. This prevents linear pathways and enables the simulation of molecular trajectories more realistically. The molecule's position, rotation and the calculated movement vector are passed to the collision detection algorithm, to find a collision with a protein. Since proteins are much larger than the reactants, a collision has no influence on their movement. On the other hand, the movement vector of a reactant is shortened in case of a collision. If a reactant needs to enter a specific protein to perform a reaction, collision between the protein and the reactant is ignored.

Reactions are processed when all included molecules are colliding. When a reaction is executed, the reactants are deleted and the created products are placed at the location of the reaction. To minimize the number of molecules stored in a compute buffer, the buffer positions of deleted molecules are saved. This way, whenever a product is created, deleted molecules can be overwritten.

For load balancing, not all molecules are included in the reaction system. Instead, the simulation area is reduced to a spherical compartment, which is placed by the user at the location of an arbitrary protein. Only reactants inside of the compartment are included in the simulation process, while the others are moved by Brownian motion.

3.2 Compartment Simulation

Since the reaction system and the collision detection algorithm are processed in real time, three techniques are used inside of the simulation compartment to reduce the number of overall calculations during simulation. By using spatial subdivision, counting sort and the fast fixed-radius nearest neighbor algorithm, the spatial position of individual objects can be included during the processing of molecular interactions.

Spatial subdivision is an approach where objects in a three dimensional space are ordered by their position. The space is partitioned in a uniform grid, such that a cell is at least as large as the largest object [Nygu07]. The objects are sorted with their corresponding cell ID by using the counting sort algorithm. Additionally, an array called Bin-Counter is used to keep track of the number of objects inside of every single cell. The Bin-Counter in combination with the list of ordered objects allows to identify all objects contained in a specific cell by the cell index.

Fast fixed-radius nearest neighbors is an algorithm to find all objects inside of a sphere with a radius R. When the sphere is centered at the position of a specific object and the radius corresponds to the length of the movement vector, the algorithm can be used to find all relevant objects during collision detection. Since the time complexity of a brute force attempt to find all neighbors of all objects is $O = (n^2)$, the spatial partitioning method is applied first. To minimize the number of cells who are overlapping with the sphere without having too many objects per cell, an additional requirement is established, which states that the minimum cell size during spacial partitioning has to be at least as large as the radius R. This way, only objects in neighboring cells have to be searched, which reduces the average complexity to $O(n * log(n))$ [WSNT]. Those objects can easily be found by combining the spatially sorted objects and the Bin-Counter values.

4 VISUALIZATION

While the technique to represent molecules in a level-of-detail manner proposed by Le Muzic et al.[MPSV14] is inherited, three additional visualization techniques, called real-time glow, cone clipping

and reaction tree, are implemented to communicate knowledge of complex biochemical processes in cells. Since the reactions are distributed throughout the compartment and can occur simultaneously, it is difficult for the user to realize when and where a reaction is completed. Therefore, every created product and proteins included in a completed reaction are highlighted with a real-time glow effect for approximately one second. The glowing objects are copied in a separate texture with a compute shader and blurred by using a two-step operation called a *separable convolution* [Fer04]. This way, the two-dimensional convolution kernel is divided into two separate one-dimensional convolutions, one in each axis, which greatly reduces the computation costs [Fer04]. To increase the brightness of the glow effect, a non-uniform convolution kernel is used.

When illustrating dense scenes with millions of atoms placed near each other, reaction processes are easily covered by larger protein structures. With the cone clipping method, the user is able to remove disturbing protein structures, which are located between the camera and the center of the simulation compartment, to get a better view of the ongoing reactions and molecular interactions during simulation. To allow the user to change the amount of clipped objects, the cone angle can be set interactively to a value between 1 and 89 degrees.
While increasing the visibility of simulation participants is the main goal of this visualization technique, the three dimensional spatial depth impression should be retained. Additionally, a semi-transparent area is used to create a continuous transition between the clipped area inside of the cone and the shown objects outside of it. Thus, a second cone is implemented and placed at the same position as the clipping cone, whereby the angle of the second cone is twice as large as the angle of the clipping cone. Objects located between the inner and outer cone are represented partially transparent. To create a continuous transition between the clipped area and the opaque objects, the amount of transparency for a specific object depends on the location. While molecules located next to opaque objects are rendered with less transparency, the value increases when located closer to the clipping cone.

In this project, a hierarchical structure is used to illustrate biological networks created by complex interactions between different molecules and proteins in a particular time span. Starting with only one reaction, a dynamic tree structure is build by illustrating the path of the reaction products with lines. When those molecules are included in another reaction, a node at the location of the reaction is included in the structure, while the outgoing branches are connected with the new products. Therefore, the tree structure is growing over time, showing the influence of the starting reaction on the whole simulation. The tree structure is created during post-rendering by using a geometry shader. This enables to visualize dynamic lines for reactants and products of ongoing reaction and the static structure created by already processed reactions.

1: **Input**: $diff$, *rm* and *sumRadii*.
2: **Output**: The new vector length or -1.
3: float $dist = length(diff)$
4: float $cd = dot(normalize(rm), diff)$
5: **if** $cd <= 0$ **then**
6: **return** -1
7: float $f = dist * dist - cd * cd$
8: float $sumRadiiSquared = sumRadii * sumRadii$
9: **if** $f >= sumRadiiSquared$ **then**
10: **return** -1
11: float $t = sumRadiiSquared - f$
12: float $newMovementLength = cd - sqrt(t)$
13: **return** $newMovementLength$

Algorithm 1: Collision Detection

5 IMPLEMENTATION

In this section, we are going to discuss implementation details about the used data structure and the collision detection algorithm.

5.1 Data Structure

Data objects are stored in a two-parted data structure to enable efficient data access and to minimize the needed amount of storage. Passive data contains all information needed to render molecules and is uploaded to the GPU when a scene is loaded and when the reactants are placed. Structural details of the molecular types, such as the atom count, the position of single atoms and the color are stored as passive data, as well as the position and rotation of every single molecule. Active data on the other hand stores the following structural information needed for the simulation process:

- *Reactant*: Is created for every reactant participating the simulation process and contains a reference to the passive data, a specific reaction, the movement vector and the cell id of the compartment.
- *Reaction*: Is created for every initiated reaction and contains a reference to all reactants, the calculated reaction position, which products are created and if the reaction is part of the reaction tree.
- *Collision*: A reference to the colliding objects and their movement vectors.
- *Reaction Tree-Line*: The start and end point and the molecular color.

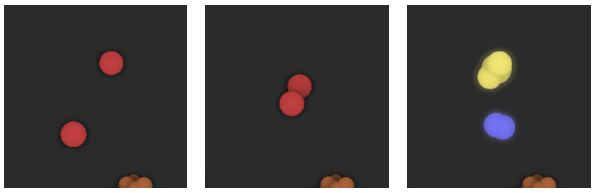


Figure 1: Snapshots from the processed reaction $H + H \rightarrow H_2 + H_2O$. (a) The two reactants approach each other. (b) The objects are colliding and the reaction is processed. (c) The reactants have been removed and the products are placed.

5.2 Collision Detection

Our algorithm is structured in a two-phase approach: a broad phase followed by a narrow phase [Nygu07]. During the broad phase, collision between molecules is determined by using minimal bounding spheres. Those tests are fast and cheap to calculate. However, since approximated bounding volumes are too imprecise for more complex shapes, errors occur in the collision tests. Therefore, potentially colliding molecules must be further processed in the narrow phase. By testing collision between two objects precisely, the single atoms of both objects have to be tested against each other. Since both phases are comparing spherical objects the same test algorithm, shown in Algorithm 1, can be used.

The algorithm calculates if two objects will collide during the next movement step and shortens the movement vector when necessary, so that both objects come to rest just at the point of contact. As input, the relative movement vector $rm = movement1 - movement2$, the spatial distance *diff* and the sum of the radii *sumRadii* are used. The radius refers to the minimum bounding sphere in the broad phase and to the atom radii during the narrow phase. The closest distance *cd* between the objects during the movement is calculated in line 4. For $cd <= 0$, the objects are not moving towards each other and no collision can occur. When the squared length of the closest distance vector is larger than the square of the summed radii compared in line 9, the objects are not colliding during their movement. At last, the Pythagorean theorem is used to shorten the movement vector in line 12 and the results are returned. After the narrow phase, the shortest calculated movement length of all pair-wise atom tests is used for the next movement step.

6 RESULTS

In this section we present results of the proposed algorithm regarding the quality of the created images and the performance of the simulation system. The simulation system is tested with a dataset created with the cellPACK [JAAGS15] modeling tool, showing a human blood serum surrounding HIV virus. Additionally, 50000 reactants of five different molecular types have been placed throughout the scene. By using a compartment radius of 300 nm, about 350 reactants have been included in the reaction system, processing four reaction types.

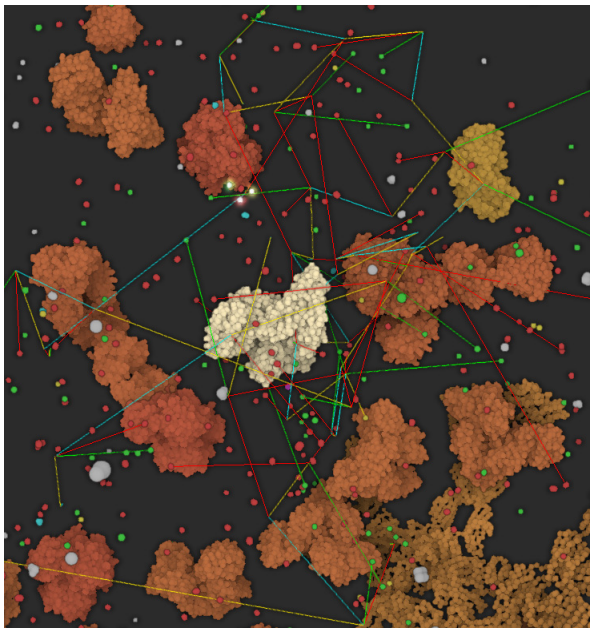


Figure 2: Snapshots from a reaction tree structure while simulating four different reaction types with approximately 350 reactants.

Figure 1 shows snapshots from the reaction process of the type $H + H \rightarrow H_2 + H_2O$. In Figure 1(a) the two reactants of type A, which are included in the reaction, are moving to the calculated reaction location. The collision of the molecules is shown in Figure 1(b). Since reactions are triggered by contact, the reaction system removes both reactants and creates the products B and C. Figure 1(c) shows the placement of the newly created products. To avoid overlapping molecules, the collision detection algorithm is used to find free areas around the reaction location.

A screenshot from the reaction tree visualization technique where the reactants are processed by four reaction types is shown in Figure 2. After initiating the starting reaction it took approximately 20 seconds of the simulation time until the tree structure has reached the illustrated size. It can be seen that created products are included in further reactions, which are initiated afterwards. By using the color of a molecular type for single lines it is possible for the user to determine which reaction has been processed at a specific location and which products have been included.

An example of the real-time glow effect is given in Figure 3. The camera is positioned in such a way that the entire compartment is shown. By highlighting the created products, the user is able to determine when and where a reaction is processed during the simulation. In Figure 3(b), a close-up of individual glowing molecules

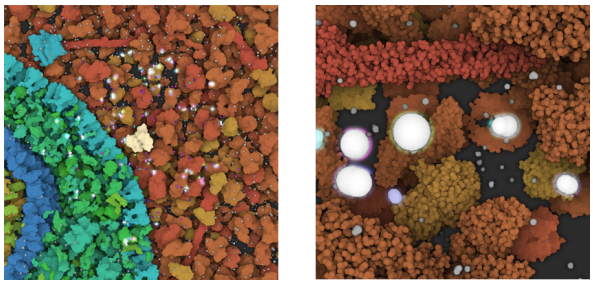


Figure 3: Snapshots from the processed reaction $A + A \rightarrow B + C$. (a) The two reactants approach each other. (b) The objects are colliding and the reaction is processed. (c) The reactants have been removed and the products are placed.

is shown. By comparing the glow radius in Figure 3(a) and (b), it can be seen that the size of the glowing effect depends on the distance of the camera to the specific object. When the camera is close to the highlighted product, the size of the glow effect around the molecule is reduced to avoid superposition of molecular structures.

A screenshot of the cone clipping effect with a 15° angle is given in Figure 4. For the simplified transparency effect, the color of a translucent object is combined with the color of concealed opaque objects or the background color, but not with other transparent objects. Although, excluding translucent objects during the alpha blending process increases the rendering performance, artifacts are created when molecules are located at the edge of the scene. An example of those artifacts is shown in the top right corner of Figure 4. When no opaque objects are located behind a transparent protein, the object color is mixed with the background color black. This leads to a wrong perception of depth, because transparent objects which are located closer to the clipping cone are shown darker than the objects behind it, which are positioned further away.

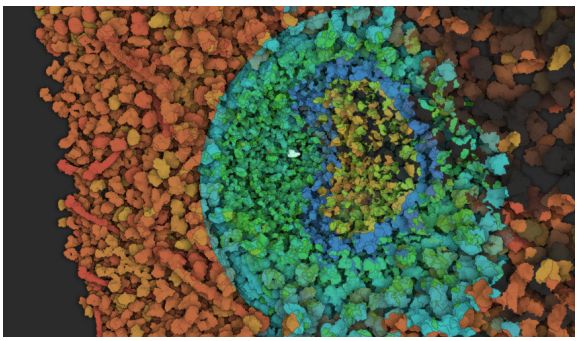


Figure 4: Snapshots from a reaction tree structure while simulating four different reaction types with approximately 350 reactants.

Performance Analysis

Since the implemented visualization techniques require a processing time of approximately 1*ms*, only the performance of the simulation system is discussed in this section. The performance of two simulations, containing 322 and 1138 reactants, has been measured on an Intel Core i7-3930 CPU 3.20 GHz coupled with a GeForce GTX Titan X graphics card using the Unity3D profiler. The results of both tests are shown in Table 1. While reactions are generated and executed when needed, only the movement step is processed per frame. Therefore, the average processing time during approximately 130 frames was measured in both tests to give a more realistic representation of the overall processing time. This time interval of about 130 frames corresponds with the adjusted reaction cycle of the COPASI API, in which new reactions are initiated. Since the reaction system stores the list of new reactions given by COPASI and processes them over time by initiating only 20 reactions per frame, the processing time of the reaction generation step fluctuates between 0ms and approximately 160ms. While the processing time of the movement step was consistent in both tests, the reaction execution step fluctuated as well. During the first test the performance to execute completed reactions was in a range between 0ms and 2ms. Since more reactions have to be executed over time with an increased number of reactants participating in the simulation system, the processing time of the reaction execution step increased and fluctuated between 0ms and 15ms during the second test. Those immense fluctuations of the processing time during reaction generation and execution lead to non-stable frame rates and possible stuttering during the simulation.

Simulation Steps	Test 1 [ms]	Test 2 [ms]
Reaction Generation	3	12
Movement	13	33
Reaction Execution	1	7

Table 1: Average performance results of the single steps of the reaction system during approximately 130 frames. The first test included 322 reactants and was executed at 45 frames per second, while the second test contained 1138 reactants and was performed at 18 frames per second.

7 CONCLUSION AND FUTURE WORK

We have introduced a tool to simulate and visualize biochemical reactions and biological networks in a large and complex multiscale structural model. Due to the collision detection algorithm, which is able to work at the level of single atoms, the implemented reaction system is able to simulate molecular interactions. For load balancing, advanced GPU programming was used for

data manipulation as well as optimization algorithms to minimize the number of calculations per frame and to enable simulation with more than 1000 reactants participating in the reaction process. Due to the size and complexity of cells and their inner life, containing billions of atoms, it is necessary to visually communicate the proceedings during the simulation to the user. Therefore, three visualization techniques have been implemented. A real-time glow effect in combination with a conical clipping object are used to point out interesting areas where reactions occur. The third implemented visualization technique, a hierarchical structure called reaction tree, is used to illustrate a biological network, by illustrating the impact of one reaction on the entire reaction system.

Although this reaction system is able simulate more realistic molecular behavior it also encloses some limitations: Firstly, the explanatory visualization of biochemical processes is limited by the overall amount of ongoing reactions during the simulation. Currently, it is not possible that the user can follow a specific molecule during the reaction process. Therefore, a combination of a leaded camera and a slow motion technique could be implemented to improve the way how the processing of individual reactions are communicated to the user. Furthermore, this approach would allow the user to follow a specific molecule through out the scene and to illustrate the molecule's reaction pathway. The second limitation refers to the simplicity of the reaction process. In CellPathway, an arbitrary protein can be assigned to single reactions. Thus, the reaction position is moved to the location of the specific protein, whereby the molecular type of the protein is not taken into account. Furthermore, proteins are not synthesized or broken apart during the reaction process. In further versions, a more advanced reaction system could be implemented to simulate changes of the cell structure. Finally, the simplicity of the visualization tree makes it difficult for the user to analyze the created structure. Therefore, additional information like the direction of a molecule's path could be visualized by arrows or a color transition of individual lines.

SUPPLEMENTARY MATERIAL

A video showing the basic functionality of the project can be found at `https://youtu.be/dKFYqH1RNW0`. The source code of CellPathway is publicly available at `https://github.com/UnityDevTeam/CellPathway`.

ACKNOWLEDGEMENTS

This work was supported through grants from the Vienna Science and Technology Fund (WWTF) through project VRG11-010 and by the EC Marie Curie Career Integration Grant through project PCIG13-GA-2013-618680.

REFERENCES

[MPSV14] Le Muzic M., Parulek J., Stavrum A.K., Viola I. Illustrative visualization of molecular reactions using omniscient intelligence and passive agents. *Computer Graphics Forum*, Vol.33, No.3, pp.141–150, June 2014.

[HSG+06] Hoops S., Sahle S., Gauges R., Lee C., Pahle J., Simus N., Singhal M., Xu L., Mendes P., Kummer U. Copasi - a complex pathway simulator. *Bioinformatics*, Vol.22, No.24, pp.3067–3074, 2006.

[KMP10] Kubera Y., Mathieu P., Picault S. Everything can be agent! *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems*, Vol.1, pp.1547–1548, 2010.

[MAPV15] Le Muzic M., Autin L., Parulek J. and Viola I. cellVIEW: a Tool for Illustrative and Multi-Scale Rendering of Large Biomolecular Datasets. *The Eurographics Association*, 2015.

[JAAGS15] Johnson G. T., Autin L., Al-Alusi M., Goodsell D. S., Sanner M. F., Olson A. J. cellpack: a virtual mesoscope to model and visualize structural systems biology. *Nature methods*, Vol.12, No.1 , pp.85–91, 2015.

[WSUnity] Unity Technologies. https://unity3d.com/. Accessed: 2016-02-11.

[WSPdb] PDB. Protein data bank contents guide: Atomic coordinate entry format description version 3.30. ftp://ftp.wwpdb.org/pub/pdb/doc/format_descriptions/Format_v33_A4.pdf. Accessed: 2016-02-11.

[CKMK13] Ciechomski P.H., Klann M., Mange R., Koeppl H. From biochemical reaction networks to 3D dynamics in the cell: The ZigCell3D modeling, simulation and visualisation framework. *Biological Data Visualization (BioVis)*, IEEE Symposium, pp.41–48, 2013.

[WSSbgn] SBGN wbsite. http://www.sbgn.org/Main_Page. Accessed: 2016-02-11.

[D14] Gehrer Daniel. CellUnity - an Interactive Tool for Illustrative Visualization of Molecular Reactions. *Institute of Computer Graphics and Algorithms, University of Technology*, 2014.

[SB01] Stiles, Joel R. and Bartol, Thomas M. and others. Monte Carlo methods for simulating realistic synaptic microphysiology using MCell. *Computational neuroscience: realistic modeling for experimentalists*, CRC Press, Boca Raton, FL, pp.87–127, 2001.

[WSCB] CellBlender website. https://code.google.com/p/cellblender/. Accessed: 2016-02-11.

[WSB] Blender Online Community. Blender - a 3d modelling and rendering package, http://www.blender.org. Accessed: 2016-02-11.

[MWPV15] Le Muzic M., Waldner M., Parulek J., Viola I. Illustrative Timelapse: A Technique for Illustrative Visualization of Particle-Based Simulations. *Visualization Symposium (PacificVis)*, 2015 IEEE Pacific, Vol., pp.247–254, 2015.

[GKMRE14] Grottel S., Krone M., Müller C., Reina G., Ertl T. MegaMol - A Prototyping Framework for Particle-Based Visualization. *Visualization and Computer Graphics*, IEEE Transaction, Vol.21, No.2, pp.201–214, 2014.

[HGVV16] Hermosillaa P., Guallarb V., Vinacuaa A., Vazqueza P.P. High quality illustrative effects for molecular rendering. *Computers & Graphics*, Vol.54, pp.113–120, 2016.

[Nygu07] Nguyen H. GPU Gems 3, Chapter 32. *Addison-Wesley Professional*, 2007.

[WSNT] Nvidia - GPU Technology Conference. http://on-demand.gputechconf.com/gtc/2014/presentations/S4117-fast-fixed-radius-nearest-neighbor-gpu.pdf. Accessed: 2016-02-11.

[Fer04] Fernando R. GPU Gems, Chapter 21. *Addison-Wesley Professional*, 2004.