*ISSN 1213-6972* *Volume 24, Number 1, 2016*

# Journal of WSCG

*An international journal of algorithms, data structures and techniques for computer graphics and visualization, surface meshing and modeling, global illumination, computer vision, image processing and pattern recognition, computational geometry, visual human interaction and virtual reality, animation, multimedia systems and applications in parallel, distributed and mobile environment.*



Vaclav Skala – Union Agency

*Journal of WSCG*

Editor-in-Chief: Vaclav Skala
c/o University of West Bohemia
Faculty of Applied Sciences
Univerzitni 8
CZ 306 14 Plzen
Czech Republic
http://www.VaclavSkala.eu

Managing Editor: Vaclav Skala

Printed and Published by:
Vaclav Skala - Union Agency
Na Mazinach 9
CZ 322 00 Plzen
Czech Republic

Hardcopy: ***ISSN 1213 – 6972***
CD ROM: ***ISSN 1213 – 6980***
On-line: ***ISSN 1213 – 6964***

# Journal of WSCG

## Editor-in-Chief


### Vaclav Skala

c/o University of West Bohemia
Faculty of Applied Sciences
Department of Computer Science and Engineering
Univerzitni 8
CZ 306 14 Plzen
Czech Republic

http://www.VaclavSkala.eu
Journal of WSCG URLs: http://www.wscg.eu or http://wscg.zcu.cz/jwscg


## Editorial Advisory Board
## MEMBERS

# 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)
GuÄŠrin,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)

# Journal of WSCG

# Vol.24, No.1, 2016

# Contents

# Coherent Metropolis Light Transport on the GPU using Speculative Mutations

Martin Schmidt
martin.schmidt@uni-bayreuth.de

Oleg Lobachev
University Bayreuth,
AI5: Visual Computing
Universitätsstr. 30
95447 Bayreuth, Germany
oleg.lobachev@uni-bayreuth.de

Michael Guthe
michael.guthe@uni-bayreuth.de

## ABSTRACT

The Metropolis Light Transport algorithm generates physically based images with superior image quality than classical ray tracing. Although it is trivially parallelizable on GPUs by running $N$ MLTs, the performance on current graphics hardware is below par. One of the main problems is the set of incoherent paths due to the independent Markov chains. Since each MLT generates full paths and mutates them sequentially, we construct totally incoherent rays which in negatively affects the performance on the GPU. By using a novel speculative variant of the Metropolis algorithm we increase the similarity of paths and achieve higher coherence. This decreases the computation time significantly. Further, we improve memory access by optimizing the data layout to better utilize coalesced access.

## Keywords

global illumination, parallel algorithms, markov chain monte carlo

## 1 INTRODUCTION

Todayâs standards in generation of high-quality images are based on ray tracing methods. Classical ray tracing [Kaj86] and its extensions have the ability to generate high-quality images with physically correct lighting and shading. Due to the algorithms nature, only visible results are calculated. In contrast to brute-force rasterization approaches, Ray tracing depends only logarithmically on scene complexity [WPS+03].

The emergence of massively parallel computing devices with commodity graphics hardware has led to increased research in the field of real-time raytracing. Modern GPUs can handle several hundred threads simultaneously. Since several years, the performance of commodity hardware (e.g. Desktop PCs) is suitable enough for real-time raytracing approaches [RSH05].

Extending classical ray tracing, the path tracing approach leads to significant improvements of visible image quality due to its physically-based rendering approach. Especially bi-directional path tracing and the Metropolis Light Transport (*MLT*) algorithm have build upon the ray tracing idea [Vea97, VG97]. These techniques behave highly parallel due to the independent nature of simultaneously traceable rays [PH10].

We show that the one of the main problems of *MLT* on the GPU, the highly incoherent path samples, can be mitigated by a different parallelization strategy. In addition to performing $N$ independent *MLTs*, we propose a speculative mutation algorithm. This reduces the

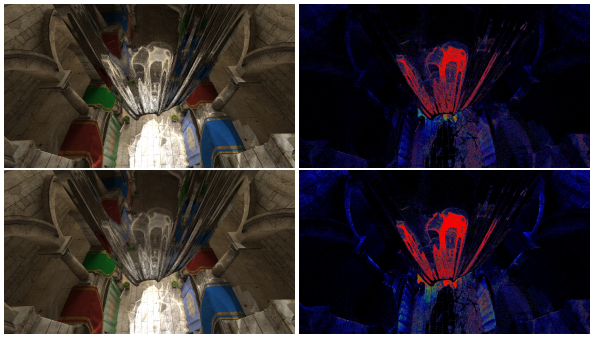


Figure 1: Equal time comparison (10 minutes) of our speculative Metropolis Light transport (top) with a naive parallelization (bottom) running on a GeForce GTX Titan. The error images denote high errors in red, medium in green and low in blue. Our approach clearly improves the convergence rate and reduces the overall error. The image was generated with a speculative tree depth of three.

overall incoherence of candidate paths and leads to a reduced divergence and better memory access patterns inside the warps and therefore a higher overall performance.

Figure 1 shows an equal time comparison on a GeForce GTX Titan after 10 minutes with a resolution of $1920 \times 1080$. For the speculative MLT, a tree depth of 3 was used. The overall convergence rate is improved not only by the fact that more mutations per pixel are performed, but also since each Markov process performs three steps in a single iteration.

The main contribution of this paper is an efficient parallelization of the *MLT* algorithm on the GPU using CUDA. In the following, we review existing work in the area of GPU-based real-time ray and path tracing (chapter 2), give a short recap on classical *MLT* (chapter 3), describe how the *MLT* can be parallelized using speculative mutations (chapter 4 and 5), show results from our experiments (chapter 6) and give a short outlook on limitations and further possible improvements (chapter 7).

## 2 RELATED WORK

With the advent of programmable graphics chips, global illumination algorithms were implemented on the GPU. The first GPU-based path tracer was published in 2002 by Purcell et al. [PBMH02]. Much work has been done since these times, including topics like efficient ray traversal [AL09, AK10, Gut14] and ray compaction or sorting [GL09].

Implementing the path tracing algorithm on the GPU has lead to new problems. Since path tracing, Bi-Directional Path Tracing and the Metropolis Light Transport (*MLT*) make use of Monte Carlo sampling methods, their behavior is stochastic in nature. Stochastic sampling lets rays terminate after different path lengths and therefore leads to incoherencies in the workload of each SM on the GPU. This was partially solved by restarting terminated rays [NHD10]. Incoherent branching and memory access lead to reduced performance on modern GPU hardware [ALK12].

Segovia et al. analyzed possibilities to adapt the *MLT* algorithm for SIMD execution on general purpose CPUs [SIP07]. They found out that proposing multiple sub path mutations at once during the mutation step increased parallel SIMD execution feasibility. They implemented the Multiply-Try Metropolis algorithm (*MTM*) [LLW00] to generate a bunch of $m$ mutation candidates during the mutation step for a given MLT sample. These candidates form sub paths for the sampled MLT path that are highly coherent. On the GPU this might also look promising since tracing coherent rays better utilizes the current hardware architecture [ALK12]. The time required per mutation however doubles due to the additional reference set. In addition, each of these paths only contributes a smaller fraction to the image, i.e. $\frac{1}{m}$ on average.

The traditional *MLT* implementation, described by Veach [Vea97, VG97], directly mutates the path vertices. While this might seem straightforward, the implementation is very complicated. Therefore, Kelemen et al. [KSKAC02] proposed to define the Markov process in terms of the random numbers that would be used in bi-directional path tracing. Despite importance sampling, this approach produces images with slightly lower quality than the original MLT. An alternative strategy is described by Hachisuka et al. [HKD14]. By combining *Markov Chain Monte Carlo (MCMC)* sampling with *Multiple Importance Sampling (MIS)*, they further decrease variance and achieve results of similar or even better quality than the original MLT. In our work, we use this combination as it has several advantages due to the way it generates samples.

## 3 METROPOLIS LIGHT TRANSPORT

The original *Metropolis Light Transport* algorithm [Vea97] extends the idea behind *bi-directional path tracing* (BDPT). It connects paths between light sources and the virtual camera lens to calculate the energy that flows between the light source and the objects and eventually reaches the virtual sensor. While this approach is similar to BDPT, the way paths are generated is different.

MLT first generates a path $\bar{x}$ that starts at the light source. The path then is extended by a series of vertices $x_0, x_1, \ldots, x_k$ for a length $k \geq 1$ [VG97]. Each vertex lies on an arbitrary surface and receives energy from the light source through the path. The exact direction for a new path segment is calculated using Monte Carlo sampling. Once a path that successfully connects the light source and the virtual sensor is found, it serves as a starting point for random path mutation.

During path mutation, the algorithm generates a *Markov chain* of path mutations $\overline{X}_0, \overline{X}_1, \ldots, \overline{X}_i$. Each mutation $\overline{X}_i$ is the result of a random walk permutation of one of the vertices of the direct predecessor $\overline{X}_{i-1}$. The mutation strategy only uses the direct predecessor as a start for the next mutation, ignoring all earlier mutations. Each new mutation is the result of a *Metropolis-Hastings* sampling in local path space.

For each mutation $\overline{X}'_i$, an *acceptance probability* function $a(\bar{y}|\bar{x})$ evaluates the chance that $\bar{y}$ will be $\overline{X}'_i$ if $\bar{x} = \overline{X}_{i-1}$. If $\overline{X}'_i$ is accepted, it becomes $\overline{X}_i$, otherwise $\overline{X}_{i-1}$ will be kept.

After the mutation has finished (either with acceptance or not), the contribution of the new path is calculated and added to the corresponding pixel in the *image plane*. Each pixel is sampled by a number of $n$ mutations, with example values of 250 [VG97].

The main advantage of the MLT is the local behavior of the mutations. Once a path with a certain contribution has been sampled, each mutation will at first be in the neighborhood of the original path. This increases performance and quality for problematic scenes where light has to travel mostly indirect, for example the Sponza scene where the light source is outside the atrium and only can enter through the open ceiling.

When trying to implement a parallel MLT algorithm, the main problem of the classical MLT is the fact that it randomly walks through path space. So when running $N$ MLTs in parallel, they usually sample completely different areas of the path space at each point in time. This is especially problematic for GPU-based implementations as coherent processing is critical to maintain high performance [AL09].

## 3.1 Primary Sampling Space MLT

Kelemen et al. [KSKAC02] proposed a novel mutation strategy for MLT path mutation that operates in the *primary sampling space* (PSSMLT). The basis is a bidirectional path tracer for which the random numbers are generated by a Markov chain. Thus the state itself is a set of random numbers that were used to generate the path. This can be seen as a single point $\overline{u}_i$ in a high dimensional – or possible infinitely dimensional – space. The mapping is shown in Figure 2.

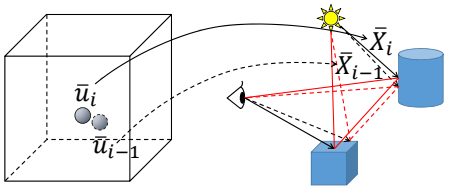



Figure 2: Primary sampling space MLT. Every multi-dimensional sampling point $\overline{u}_i$ is mapped to a path $\overline{X}_i$. Similar points map to similar paths.

Mutations are thus simply local or global movements of this point, where large changes – i.e. generating new random numbers – correspond to the original BDPT. This has three advantages. First, any multi-dimensional random number produces a valid path, where directly changing the path vertices often produces invalid paths. Second, similar sample points produce similar paths which can be used to control the magnitude of path change. And finally, the mutations can easily be made symmetric which removes one of the more complicated terms for computing the acceptance probability.

While all these significantly ease the implementation and reduce the memory that is needed to store the path information, the resulting quality is slightly lower than that of the original MLT. One of the reasons is that BDPT connects the eye and light path at all vertices. In combination with the fact that very often the vertices at the end of the eye and light subpath cannot be connected, this means that the same contribution would have been made by a shorter path that had required less computation time.

## 3.2 Multiplexed MLT

An improvement of this approach is the *Multiplexed MLT* (MMLT), where the mapping from the random numbers to path space is not unique. Instead of connecting all vertices, only the end points of the eye and light path are connected. For a fixed path length of $k$, there are thus $k+2$ possible mappings of the random numbers to a path. The eye path length needs to be chosen between 0 to $k+1$ vertices and the light path contains the remaining vertices. This is done using a tempering parameter $t$ that is also selected using the same Markov chain as for the random numbers. The mapping from $(\overline{u},t)_i$ to $\overline{X}_i$ is shown in Figure 3.

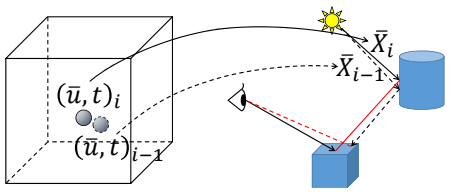



Figure 3: Multiplexed MLT. An additional tempering parameter $t$ is used to determine the length of the eye and light subpath. Only the end points are connected.

In contrast to the PSSMLT, every edge that was traced is used to calculate the contribution. If the end points cannot be connected, e.g. because one of the surface normals points away from the connecting edge, the path contribution becomes zero. This means that such a path is never accepted and the MLT continues its search from the previous path $\overline{X}_{i-1}$ that had a contribution. The same holds for a path with occluded connecting edge. Thus mostly paths where every edge transports energy are sampled. This means that the number of unnecessarily traced rays is reduced compared to PSSMLT. In the end, the method produces results that are competitive with the original MLT with respect to peak signal noise ratio. The simple mapping from multi-dimensional points to valid paths has however a significant advantage as we will discuss in the following.

## 3.3 Problem Statement

Both of these approaches share the property that similar random numbers generate similar paths. Thus the coherence between neighboring threads that handle different Markov chains can be increased if they are based on similar random numbers. Simply using $N$ independent Markov chains however leads to samples that are distributed throughout the sampling space.

One possibility would be to sort the paths according to their random numbers but this adds a significant overhead that is not easily alleviated by the improved coherency. Using Multiple-Try mutations also generates

similar paths but adds the overhead of generating a reference set and reduces the contribution of each path. If $m$ tries are generated, the contribution of each is expected to be $\frac{1}{m}$. Another possibility would be to generate a set of mutations from the current state and successively test them until the first one is accepted [BJB10]. While this reject chain (RC) sampling is a good strategy for Markov Chains with low acceptance probability, all MLT variants try to achieve an acceptance rate close to 1.

If we use a PSSMLT or MMLT, the mutation itself does not depend on the actual path but only on the random numbers that define it. Thus we can generate more than a single mutation – and their corresponding paths – in parallel. Instead of only sampling the reject chain, we can sample all possible paths up to a depth of $d$.

# 4 SPECULATIVE MLT

The main idea behind the *speculative MLT* (SMLT) is to simultaneously and speculatively evaluate possible mutations from a candidate set. To this end, we need to perform three steps: First, we generate the sampling points in primary space. Then we evaluate the corresponding paths. Finally, we accumulate the contributions and choose the final candidate.

Performing all possible mutations up to a given depth $d$ produces a binary tree of candidates. As global mutations completely change the sampling points in primary space, we only allow them as first mutation of the local tree. All other mutations are always local ones that produce similar paths. Details on the parallel implementation are discussed in section 5.1.

Then we trace the paths to compute their transported energy, probability and weight as discussed in [HKD14]. From this we can compute the local acceptance probability $a(\overline{X}_b|\overline{X}_a)$ for each mutation from $\overline{X}_a$ to $\overline{X}_b$. For a single mutation step starting at $\overline{X}_{i-1}$, we then have the following iteration:

$$\overline{X}_i = \begin{cases} \overline{X}'_i & : \quad a(\overline{X}'_i|\overline{X}_{i-1}) \\ \overline{X}_{i-1} & : \quad 1-a(\overline{X}'_i|\overline{X}_{i-1}) \end{cases} \tag{1}$$

When extending the local mutation tree to a depth of 2, we also need to consider the two possible mutations from $\overline{X}_{i-1}$ to $\overline{X}^1_{i+1}$ and from $\overline{X}'_i$ to $\overline{X}^2_{i+1}$. If $a_1$ denotes the acceptance probability from $\overline{X}_{i-1}$ to $\overline{X}_i$, $a^2_1$ from $\overline{X}_{i-1}$ to $\overline{X}^1_{i+1}$ and $a^2_2$ from $\overline{X}'_i$ to $\overline{X}^2_{i+1}$, we can write the total acceptance probabilities as:

$$\begin{aligned} p(\overline{X}_{i-1}) &= (1-a_1)\cdot(1-a^2_1) \\ p(\overline{X}^1_{i+1}) &= (1-a_1)\cdot a^2_1 \\ p(\overline{X}_i) &= a_1\cdot(1-a^2_2) \\ p(\overline{X}^2_{i+1}) &= a_1\cdot a^2_2 \end{aligned} \tag{2}$$

Note that these always sum up to $\Sigma p = 1$. Equation 2 is illustrated in Figure 4. For larger trees, the probability for each candidate is the product of all accept/reject probabilities from the root down to the leaf level.

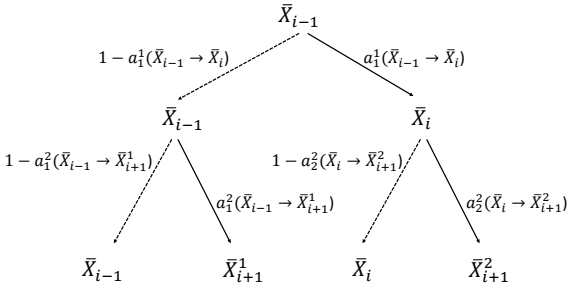



Figure 4: Mutation tree and acceptance probabilities for a mutation depth of $d = 2$.

Although we have a candidate set of $2^d$ samples, we only need to trace the path for $2^d - 1$ of them. $\overline{X}_{i-1}$ is the first-chance rejection and therefore still the same path as the result from the last iteration. By increasing the size of the candidate set, we perform $d$ iterations in parallel on a set of $\sim \frac{N}{2^d-1}$ MLTs. So in addition to more efficient tracing, we also expect a lower start-up bias and faster convergence to the stationary distribution.

## 4.1 Variance reduction

Similar to previous methods [VG97, HKD14], we want to minimize the variance of the generated image. Therefore, we accumulate the expectation value of all candidates instead of the chosen path only.

In contrast to computing the acceptance probability, we need however not only consider the leaf level of the tree as this would mean to skip all iterations except the last one. Instead, we calculate the contribution at each level of the tree, except the root node which was already accumulated in the last step. In our example with a depth of 2, the contribution weight $w$ for each path is:

$$\begin{aligned} w(\overline{X}_{i-1}) &= (1-a_1)\cdot(2-a^2_1) \\ w(\overline{X}^1_{i+1}) &= (1-a_1)\cdot a^2_1 \\ w(\overline{X}_i) &= a_1\cdot(2-a^2_2) \\ w(\overline{X}^2_{i+1}) &= a_1\cdot a^2_2 \end{aligned} \tag{3}$$

Node that these always sum up to $\Sigma w = d$, which is 2 in this example. Like in this example, the equations for deeper trees are similar to the acceptance rates. The only difference is that – with the exception of $a_1$ as discussed before – all $(1-a^i_j)$ become $(2-a^i_j)$.

# 5 IMPLEMENTATION WITH CUDA

For the implementation of the speculative MLT, we extended our existing MMLT implementation in CUDA.

The general process per iteration can be subdivided into the following parts: First, we generate the candidate mutations for the given depth $d$. Then we trace all new paths. Finally, we compute the contributions and select the surviving paths.

## 5.1 Mutation

The mutation step computes one candidate per thread. From each current path $p$, we generate a set of $2^d - 1$ candidates. Each thread loops over the variables of $p$ in the primary sampling space. For each variable $i$, we first load the corresponding one from $p$. By arranging the variables in the path buffer at position $p + i \cdot N$, we access them with a stride of 1 and partial broadcasts.

Then each thread generates a random number representing the last mutation of variable $i$. To apply the mutation to all relevant candidates, this number is stored in shared memory. Starting from the first mutation, they are applied to all relevant candidates per level using the original mutation strategy [KSKAC02]. For all mutations except the last one, the random number is fetched from shared memory. Note that this is similar to a reduction using a binary tree in shared memory. The only difference is that the accumulation is performed towards the leaves and not towards the root. Figure 5 shows this process for an example depth of 2. Finally, the new sampling space variables are again stored with stride 1 access in the larger candidate buffer.

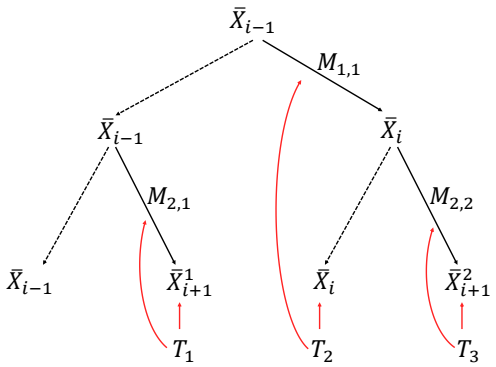



Figure 5: 3 threads are working on 3 mutations; mutation depth of $d = 2$.

## 5.2 Tracing

After generating the candidates in primary sampling space, we trace the paths using a ray scheduler based on the wavefront path tracer [LKA13]. First all eye and light segments are stored in a ray buffer. Then the intersection points for these rays are determined using a highly optimized trace kernel [Gut14]. The BRDF is evaluated at the hit points and secondary rays are constructed based on the stored *random numbers* in the candidate set. Finally, the shadow rays to connect the path ends are constructed and again traced using the same trace kernel.

While tracing will be divergent even for coherent paths, ray construction is mostly non-divergent (except for rays that exited the scene). Here again, the memory layout of the candidate set leads to a coalesced stride 1 access into the candidate buffer.

Due to the ray restarting of the trace kernel, the performance will gradually increase with the number of coherent rays. However, it will level once it reaches the warp size of (currently) 32.

## 5.3 Selection

For selection, we could again start one thread per candidate and coordinate the work over shared memory. As we still have a high amount of parallel MLTs, it is however more efficient to handle each selected path $p$ with its own thread.

While reading the sample state now has a stride of $2^d - 1$, the problem can be alleviated for the path contribution. Each candidate path $c$ stores *RGB* values for the transported energy multiplied with the weight from the balance heuristic [HKD14] and a pixel index for each pixel it contributes to. By combining this data into a single `float4`, we only need a single memory access and loose fewer bandwidth. The pixel index is then accessed using the *intrinsic* functions `__int_as_float` and `__float_as_int`.

# 6 RESULTS

We tested our implementation on an *Intel Core i7-3720* quadcore CPU with 16 GB of RAM, paired with an *nVidia GeForce GTX Titan* with 6 GB video memory. As test cases we chose three scenes listed in table 1 to evaluate different light transport scenarios.

| Scene | # Triangles | # Vertices |
|---|---|---|
| Sponza | 279,157 | 193,300 |
| Sibenik | 75,284 | 83,490 |
| Conference | 331,179 | 216,862 |

Table 1: Number of triangles and vertices per mesh.

Figure 6 shows the ground truth images of the views we used for the evaluation. These images were generated using a bi-directional path tracer with 10 million samples per pixel with a maximum of 10 bounces and the pseudo-random Halton sequence. We added a highly specular banner to the Sponza scene (cosine lobe with an exponent of $10^5$) to produce caustics and reflected caustics. These are especially visible in view 2. The Sibenik cathedral is illuminated by the light shining through the windows only, so only few paths contribute to the image. This is even more the case in the final example, the Conference scene. Here there is only indirect light coming through the sunblind with at least two bounces.

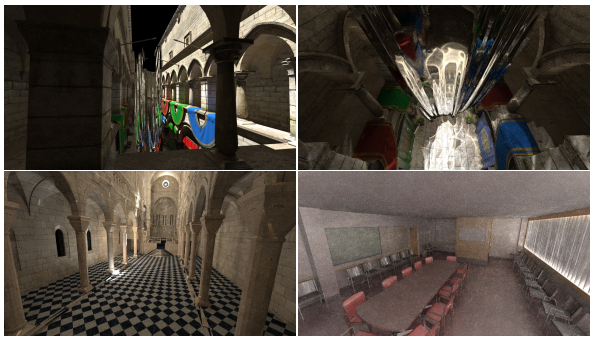


Figure 6: Scenes and views (ground truth images) used in the evaluation. From left to right and top to bottom: Sponza view 1, Sponza view 2, Sibenik and Conference.

Each scene was tested with the same view and a screen resolution of $1920 \times 1080$ for all of the methods. We compared the naively parallelized MMLT with our speculative MLT using different mutation depths and the reject chain MLT using different chain lengths. Table 2 shows the number of parallel MLTs $N$ and the total number of parallel mutations. For the *mutate kernel*, we thus launch 1024 blocks with 248 to 256 threads.

| | # MLTs (N) | # mutations |
|---|---|---|
| naive | $256 \cdot 2^{10}$ | $256 \cdot 2^{10}$ |
| S2 | $85 \cdot 2^{10}$ | $255 \cdot 2^{10}$ |
| S3 | $36 \cdot 2^{10}$ | $252 \cdot 2^{10}$ |
| S4 | $17 \cdot 2^{10}$ | $255 \cdot 2^{10}$ |
| S5 | $8 \cdot 2^{10}$ | $248 \cdot 2^{10}$ |
| S6 | $4 \cdot 2^{10}$ | $252 \cdot 2^{10}$ |
| RC2 | $128 \cdot 2^{10}$ | $256 \cdot 2^{10}$ |
| RC4 | $64 \cdot 2^{10}$ | $256 \cdot 2^{10}$ |
| RC8 | $32 \cdot 2^{10}$ | $256 \cdot 2^{10}$ |
| RC16 | $16 \cdot 2^{10}$ | $256 \cdot 2^{10}$ |
| RC32 | $8 \cdot 2^{10}$ | $256 \cdot 2^{10}$ |
| RC64 | $4 \cdot 2^{10}$ | $256 \cdot 2^{10}$ |

Table 2: Number of parallel running MLTs and the number of parallel mutations.

The results show an almost linear speedup with the depth for our speculative parallelization, compared to the naive approach of using independent MLTs. Table 3 and Figure 7 show the linear growth of the performance for both. The main reason for the speedup is the increase in coherence that directly translates into a higher trace performance. Once the set of coherent paths grows beyond the warp size of 32, the performance does not increase any more. This is as expected since packets of at most 32 rays are fetched by the ray tracer [ALK12].

In addition, we compare the peak signal noise ratio of the generated images after rendering for 10 minutes in Table 4. Depending on the scene characteristics, the best results can be achieved with $d = 2$ (Sibenik, Conference) to $d = 3$ (Sponza view 2). With increasing depth, more paths are wasted and the quality gradually drops. This is especially true for simple views, where the acceptance probability is close to 1. In such cases, e.g. view 1 of the Sponza scene, the naive parallelization produces the best results. Note that the reject chain approach never produces better images in the same time than the naive one. This is due to the fact that the contribution of the first additional candidates is almost always below $\frac{1}{4}$ and drops exponentially with further ones.

| | Sponza view 1 | Sponza view 2 | Sibenik | Conference |
|---|---|---|---|---|
| naive | 14.48 | 9.80 | 15.92 | 8.31 |
| S2 | 15.61 | 10.44 | 16.40 | 8.90 |
| S3 | 15.67 | 10.69 | 16.86 | 9.07 |
| S4 | 15.87 | 10.86 | 17.11 | 9.47 |
| S5 | 16.33 | 11.09 | 17.60 | 9.77 |
| S6 | 16.42 | 11.10 | 17.63 | 9.78 |
| RC2 | 15.49 | 10.36 | 16.41 | 8.68 |
| RC4 | 15.84 | 10.98 | 17.51 | 9.19 |
| RC8 | 16.89 | 11.40 | 18.18 | 9.65 |
| RC16 | 17.66 | 11.99 | 18.38 | 10.19 |
| RC32 | 17.67 | 12.39 | 19.24 | 10.63 |
| RC64 | 17.76 | 12.53 | 19.36 | 10.64 |

Table 3: Million mutations per second for the naive parallelization, the speculative with depth $d$ (S$d$) and reject chain with length $l$ (RC$l$).

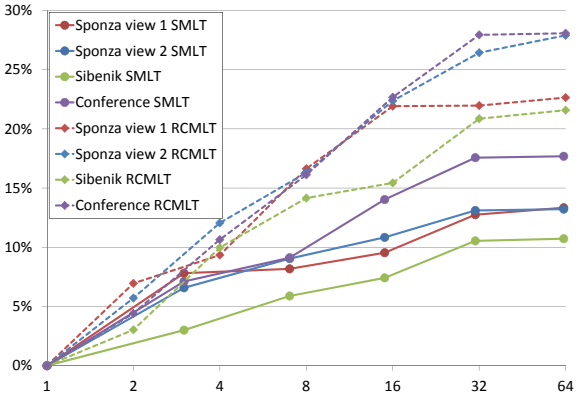



Figure 7: Relative speedup of speculative MLT (SMLT) and reject chain MLT (RCMLT) compared to naively parallelized MMLT; on the x-axis are the sizes of candidate sets.

Figure 8 compares the generated images using the best parallelization for each view with the naive approach. The error and noise are especially reduced in difficult cases. These are the reflected caustic in sponza view 2 and the indirect light illuminating the conference scene. Note that for sponza view 2, neither of them has been able to converge to the stationary distribution yet, so the reflected caustic appears slightly too dark in both images.

| | Sponza view 1 | Sponza view 2 | Sibenik | Conference |
|---|---|---|---|---|
| naive | **28.61** | 17.27 | 28.50 | 12.44 |
| S2 | 28.47 | 17.90 | **28.60** | **12.50** |
| S3 | 28.00 | **18.10** | 27.64 | 12.20 |
| S4 | 27.84 | 17.88 | 26.19 | 11.62 |
| S5 | 27.56 | 17.72 | 24.80 | 11.09 |
| S6 | 27.17 | 17.33 | 23.29 | 10.38 |
| RC2 | 28.29 | 17.09 | 26.03 | 11.59 |
| RC4 | 26.21 | 16.43 | 23.13 | 10.73 |
| RC8 | 20.50 | 15.12 | 20.24 | 9.90 |
| RC16 | 18.76 | 14.35 | 17.63 | 9.10 |
| RC32 | 17.95 | 13.99 | 15.91 | 8.43 |
| RC64 | 17.27 | 14.00 | 15.45 | 7.93 |

Table 4: Peak signal noise ratio comparison of different parallelization strategies after rendering 10 minutes. The best is marked in bold.

Figure 9 shows an equal time comparison for speculative tree depths from 1 (naive) to 6. While the error in the reflected caustic is decreasing with higher tree depth, the overall error increases due to the decreasing weights of deeper paths. This clearly shows that deeper trees are suitable for paths that are difficult to sample.

## 7 CONCLUSION AND LIMITATIONS

We have proposed a novel approach for parallelizing the Metropolis Light Transport algorithm on the GPU. Our approach successfully utilizes the graphics hardware to achieve a substantial speedup compared to naively parallelized MLT. This is mostly accomplished by evaluating paths that are more coherent. This shows significantly better performance on GPUs where divergence in both execution and memory access imposes a severe penalty.

While our approach could be extended to other Metropolis sampling algorithms, it requires that the sample generation and evaluation can be decoupled. This means that it must be possible to generate the candidate $\overline{X}'_i$ without having evaluated the previous sample $\overline{X}_{i-1}$. Therefore, our approach cannot be applied to the original MLT algorithm where the path is mutated directly.

Another problem of our approach is that the contribution of a sample exponentially decreases with the depth. This leads to an optimal depth of 2 or 3 which in turn only generated 3 or 7 coherent samples. On the other hand, paths that are difficult to sample – like reflected caustics – are better handled with a depth of 5 or even 6, i.e. 31 or 63 coherent samples. In the future we therefore plan to evaluate other sampling strategies to generate larger sample sets without reducing the contribution of some paths.

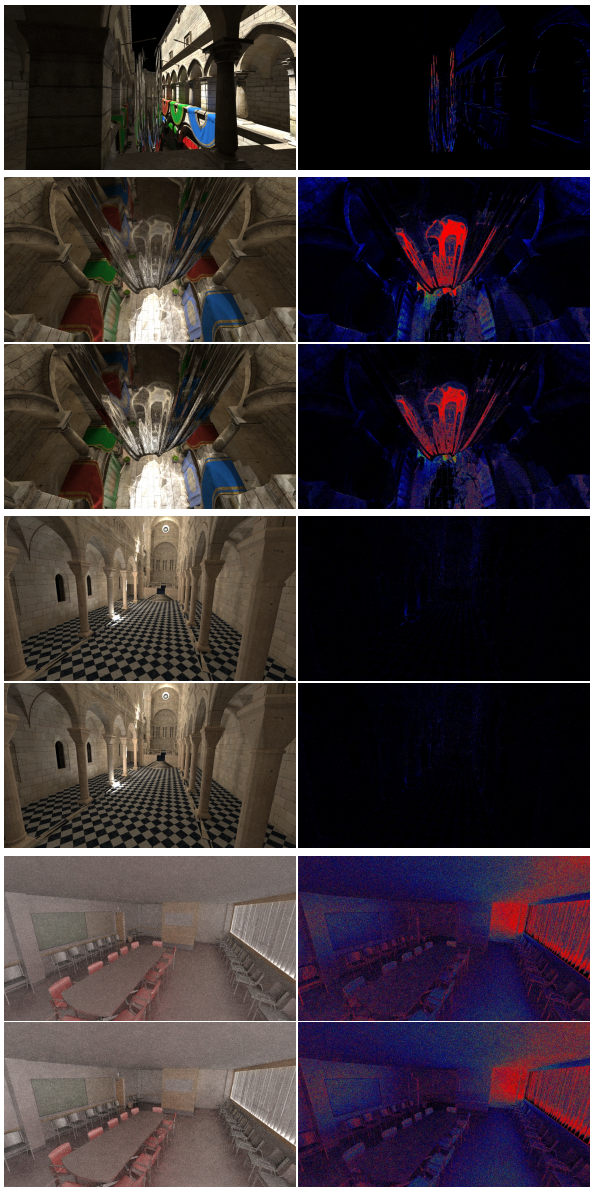


Figure 8: Equal time comparison (10 minutes on a GeForce GTX Titan) using the naive parallelization (upper image) and the best method for each (lower image). The chosen method from top to bottom is: naive, S3, S2 and S2 (c.f. Table 4). The error images show high errors in red, medium in green and low in blue/black.

## 8 REFERENCES

[AK10] T. Aila and T. Karras. Architecture considerations for tracing incoherent rays. In *Proceedings of High-Performance Graphics 2010*, pages 113–122, 2010.

[AL09] T. Aila and S. Laine. Understanding the efficiency of ray traversal on gpus. In *Proceedings of High-Performance Graphics 2009*, pages 145–149, 2009.

[ALK12] T. Aila, S. Laine, and T. Karras. Under-

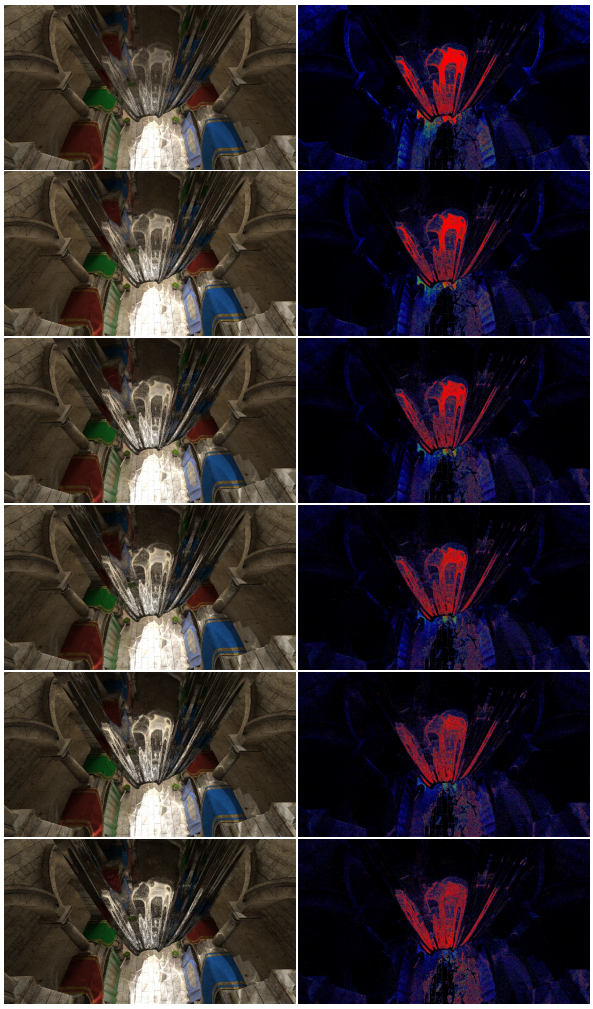


Figure 9: Sponza view 2 with different speculative tree depth $d$ increasing from 1 (naive) to 6. All images were generated in 10 minutes on the GeForce GTX Titan.

standing the efficiency of ray traversal on GPUs – Kepler and Fermi addendum. NVIDIA Technical Report NVR-2012-02, NVIDIA Corporation, June 2012.

[BJB10] J. Byrd, S. Jarvis, and A. Bhalerao. On the parallelisation of mcmc by speculative chain execution. In *Parallel Distributed Processing, Workshops and Phd Forum (IPDPSW), 2010 IEEE International Symposium on*, pages 1–8, April 2010.

[GL09] K. Garanzha and C. Loop. Fast ray sorting and breadth-first packet traversal for gpu ray tracing. *Computer Graphics Forum*, 29(2):289–298, 2009.

[Gut14] M. Guthe. Latency considerations of depth-first gpu ray tracing. In *Proceedings of Eurographics 2014 - Short Papers*, pages 53–56, 2014.

[HKD14] T. Hachisuka, A. S. Kaplanyan, and C. Dachsbacher. Multiplexed metropolis light transport. *ACM Trans. Graph.*, 33(4):100:1–100:10, July 2014.

[Kaj86] J. T. Kajiya. The rendering equation. In *SIGGRAPH '86 - Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, 1986.

[KSKAC02] C. Kelemen, L. Szirmay-Kalos, G. Antal, and F. Csonka. A simple and robust mutation strategy for the metropolis light transport algorithm. *Computer Graphics Forum*, 21(3):531–540, 2002.

[LKA13] S. Laine, T. Karras, and T. Aila. Megakernels considered harmful: Wavefront path tracing on gpus. In *Proceedings of High-Performance Graphics 2013*, 2013.

[LLW00] J. S. Liu, F. Liang, and W. H. Wong. The multiple-try method and local optimization in metropolis sampling. *Journal of the American Statistical Association*, 95(449):121–134, 2000.

[NHD10] J. Novák, V. Havran, and C. Dachsbacher. Path regeneration for interactive path tracing. *Proceedings of Eurographics 2010 - Short Papers*, pages 61–64, 2010.

[PBMH02] T. J. Purcell, I. Buck, W. R. Mark, and P. Hanrahan. Ray tracing on programmable graphics hardware. In *Proceedings of ACM SIGGRAPH 2002*, pages 703–712, 2002.

[PH10] M. Pharr and G. Humphreys. *Physically Based Rendering, Second Edition: From Theory To Implementation*. Morgan Kaufmann Publishers Inc., 2nd edition, 2010.

[RSH05] A. Reshetov, A. Soupikov, and J. Hurley. Multi-level ray tracing algorithm. *ACM Transactions on Graphics*, 24(3):1176–1185, July 2005.

[SIP07] B. Segovia, J.-C. Iehl, and B. Péroche. Coherent Metropolis Light Transport with Multiple-Try Mutations. Technical report, LIRIS UMR 5205 CNRS/INSA de Lyon/Université Claude Bernard Lyon 1/Université Lumière Lyon 2/École Centrale de Lyon, April 2007.

[Vea97] E. Veach. *Robust Monte Carlo methods for Light Transport Simulation*. PhD thesis, Standford University, 1997.

[VG97] E. Veach and L. J. Guibas. Metropolis light transport. In *SIGGRAPH '97 - Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '97, pages 65–76, 1997.

[WPS+03] I. Wald, T. J. Purcell, J. Schmittler, C. Benthin, and P. Slusallek. Realtime ray tracing and its use for interactive global illumination. Eurographics State of the Art Reports, 2003.

# Robust Object Detection for Video Surveillance Using Stereo Vision and Gaussian Mixture Model

Lars Klitzke, Carsten Koch
Hochschule Emden/Leer
University of Applied Sciences
Department of Informatics and Electronics
Constantiaplatz 4
26723 Emden, Germany
{Lars.Klitzke | Carsten.Koch}@hs-emden-leer.de

## ABSTRACT

In this paper, a novel approach is presented for intrusion detection in the field of wide-area outdoor surveillance such as construction site monitoring, using a rotatable stereo camera system combined with a multi-pose object segmentation process.
In many current surveillance applications, monocular cameras are used which are sensitive to illumination changes or shadow casts. Additionally, the object classification, spatial measurement and localization using the 2D projection of a 3D world is ambiguous. Hence, a stereo camera is used to calculate a 3D point cloud of the scenery which is nearly unaffected by illumination changes, therefore enabling robust object detection and localization in the 3D space. The limited viewing range of the stereo camera is expanded by mounting it onto a rotatable tripod. To detect objects in different poses of the camera, pose specific Gaussian Mixture Models (GMM) are used. However, changing illumination outside the current field of view of the camera or spontaneously changing lighting conditions caused by e.g. lights controlled by motion sensors, would lead to false-positives in the segmentation process if using the brightness values. Hence, segmentation is performed using the calculated point cloud which is demonstrated to be robust against changing illumination and shadow casts by comparing the results of the proposed method with other state of the art segmentation methods using a database of self-captured images of a public outdoor area.

## Keywords

Video Surveillance, Gaussian Mixture Model, Stereo Vision, Object Detection

## 1 INTRODUCTION

The usage of video systems for surveillance applications in public, industrial and private domains has been increasingly popular in the last years. Ongoing technological development led to smarter systems which enable automatic identification of critical situations or suspicious objects. The aim of those so called Intelligent Video-System (IVS) is to relieve the strain on human security guards of these systems, because they are typically monitoring multiple screens simultaneously and need to reliably detect salient behaviour. However, this is a challenging task even if they just need to monitor two screens at the same time due to the effects of fatigue and hence, inattentiveness [LCK13]. IVS can reduce the cost of video surveillance systems and increase the productivity at the same time.
The vast majority of current outdoor video surveillance systems uses monocular cameras in which the process of image segmentation is challenging. The cameras are sensitive to shadow casts of objects and (spontaneously) changing illumination conditions. Typical environments of outdoor surveillance are influenced by effects of artificial lights sources which may be (de)activated spontaneously. Additionally, the real world position and true size of detected objects cannot be determined unambiguously by using the 2D projection of the 3D world. Nevertheless, these information may be relevant for surveillance systems e.g. to locate objects in an area accurately or to track their movement in the real world. Additionally, objects can be classified using these informations as e.g. humans, animals or trees to decrease the false-positive object detection rate.

In this paper, we present an approach for intrusion detection in the field of wide area outdoor surveillance, e.g. construction site monitoring by using a multi-pose object segmentation process combined with a rotating stereo camera.

The calculated 3D point cloud of the stereo camera system is used to detect objects. The generation of this point cloud is nearly unaffected by changing lighting conditions. For segmentation, a system pose specific Gaussian Mixture Models (GMM) is used to detect objects in a wider area without false-positives caused by changing illumination outside the current field of view.

It is shown that the 3D-segmentation process by analysis of distance information instead of the brightness values, which are very sensitive to varying illumination conditions, is more robust in cases of overlapping objects, changing lighting conditions and shadow casts. The outdoor scene image datasets generated for this work have been made available online[1].

This paper is structured as following: Section 2 gives an overview of related work on camera surveillance applications. The proposed method for object detection using the calculated 3D point cloud is described in section 3. In section 4 the rotating stereo system is shown and the segmentation process is compared to other state of the art methods. A conclusion of this paper and an overview about further developments is given in section 5.

## 2 RELATED WORK

The field of camera based surveillance systems has been broadly addressed in the last decades. For instance Haritaoglu *et al.* [HHD98] presented a system called $W^4S$ which uses a stereo camera combined with an intensity based model. They show that the segmentation process is more robust in case of overlapping objects using the distance information instead of the intensity values.

Another approach was presented by Douret and Benosman [DB04]. They used a network of cameras for an intelligent traffic control system to retrieve the height of objects by assuming a plane ground model. However, Kumar *et al.* [KMP09] showed that a missing link between object and ground can lead to inaccurate position information. Therefore, they compared the result of their proposed stereo localization procedure using two pan-tilt-zoom (PTZ) cameras with a monocular approach. With the PTZ cameras they are able to determine the position of objects even when using different zoom levels and in case of overlapping objects. This is realized by using a neural network based interpolation method with an offline calculated look-up table to rectify the images online. However, compared to the rectification process of static stereo camera systems, this procedure is more computational expensive.

Nevertheless, because of the flexibility of PTZ camera and due to the need of observing even greater areas than actual static and monocular cameras can cover, much work has been done on surveillance systems using (dual) PTZ cameras [KMP09][ZWW10][ZOS13]. However, all of those systems perform the image segmentation in the 2D space and localize the detected objects in the world afterwards. The following work will present a more robust segmentation method based on the calculated distance information of the stereo camera. A detailed overview of the current state of the art of intelligent video systems is given by Liu *et al.* in [LCK13].

The main task of a video surveillance system is intrusion detection. A robust segmentation of the image into foreground and background is vital for this task. The problem of image segmentation is widely discussed in the field of image processing. Due to this, there are several methods which can be used. Sen-Ching and Kamath [SCK04] evaluate the result of Frame Differencing, Kalman Filter, Median Filter and Mixture of Gaussian (MoG) using an urban video sequence. They showed that the result of MoG (Gaussian Mixture Model (GMM) respectively), which was proposed by Friedman and Russel [FR97] and extended by Stauffer and Grimson [SG99], outperforms the other methods in many cases, e.g. outdoor surveillance.

Due to this and because of the insensitivity to local movement in the scene, e.g. swaying branches and adaptation to changing illumination conditions, the GMM is used in section 3.2, combined with a 3D point cloud. The latter is provided by a rotating stereo camera realising an even more robust segmentation, while at the same time covering a greater area than a static monocular approach without the computational complexity of handling PTZ cameras.

## 3 INTRUSION DETECTION

### 3.1 Online calibration

Typically, cameras for surveillance applications are mounted at elevated positions on walls or poles. This leads to a typical constellation as shown in Fig. 1. To directly measure the true size of detected objects, the calculated point cloud using the stereo

[1] `https://mux.hs-emden-leer.de/lkl`

camera needs to be transformed from the camera's coordinate space $C$ to the world coordinate space $W$. This transformation is used to rotate and translate the point cloud, so that the camera is virtually located on the ground level and points straight ahead. An online calibration method is used in which the user needs to select the ground of the scenery in the image.

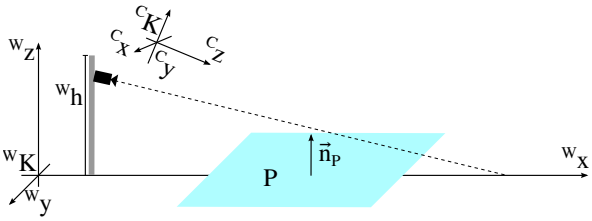



Figure 1: Model of a surveillance camera mounted at a wall or pole looking downwards.

At first, the normal ${}^{C}\vec{n}_P$ of the plane (the selected ground) in the camera's coordinate space is estimated using a method from Kovesi [Kov00] based on RANSAC. Then, the rotation ${}^{C}\boldsymbol{\Omega}_W$ between ${}^{C}\vec{n}_P$ and the vector ${}^{C}\vec{e}=[0\ 1\ 0]^T$ is calculated with Rodigues' rotation formula [Cor11]. Afterwards, the translation vector ${}^{C}\boldsymbol{\tau}_W$ of the camera relative to the ground can be estimated. Therefore, a median point $\bar{x}$ of the point cloud is calculated and rotated using (1).

$$\bar{x}' = {}^{C}\boldsymbol{\Omega}_W \bar{x} \tag{1}$$

At least, the translation vector only consists of the third part of the rotated median point (see (2)), because it represents the height of the camera in the world.

$${}^{C}\boldsymbol{\tau}_W = \begin{bmatrix} 0 \\ 0 \\ -\bar{x}'_2 \end{bmatrix} \tag{2}$$

As already stated in the beginning, the system is able to pan and tilt. Due to this, the calculated parameters become invalid if the system moves. To overcome this, the user can define specific poses $\xi_1, ..., \xi_n$ of the camera system for each of which the described calibration method needs to be performed once. Hence, a system pose $\xi_i$ is defined by (3).

$$\xi_i = \left[ {}^{C}\boldsymbol{\Omega}_W \; {}^{C}\boldsymbol{\tau}_W \right] \tag{3}$$

## 3.2 Object detection

In the following, the processing chain (see Fig. 2) for object detection is described using the previously estimated parameters for the system poses. We assume that the stereo camera is calibrated, so

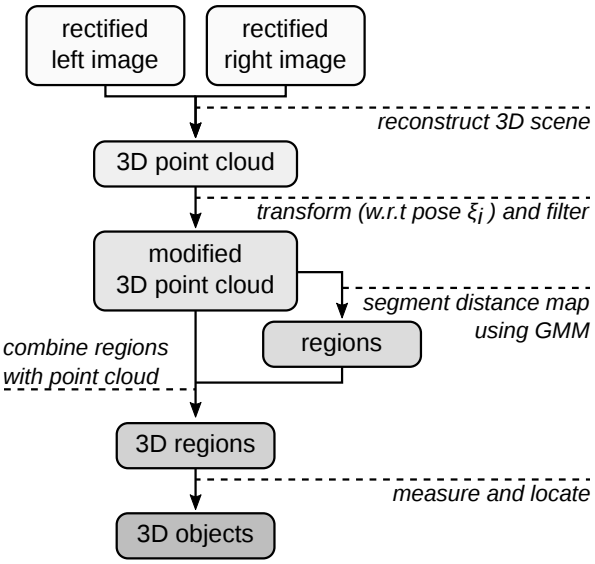



Figure 2: Overview of the processing chain for detecting objects based on the rectified stereo camera images.

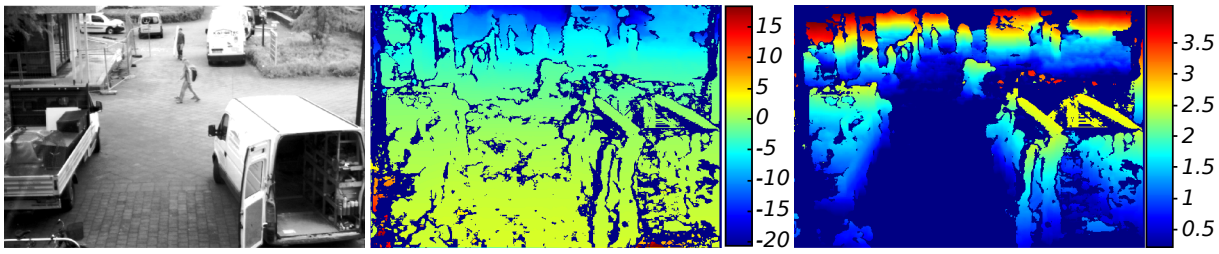
we start with the rectified images of the left and right camera. First, the disparity map has to be calculated. The Semi-Global Matching (SGM) algorithm proposed by Hirschmüller is an established method for this task. Based on the disparity map the 3D point cloud of the scenery is reconstructed. In the next step, this raw point cloud ${}^{C}\mathbf{P}$ is transformed with respect to the current system pose $\xi_i$ from (3) using (4) and the homogeneous coordinates ${}^{C}\tilde{\mathbf{P}}$ of the point cloud ${}^{C}\mathbf{P}$.

$${}^{W}\mathbf{P} = \xi_i \cdot {}^{C}\tilde{\mathbf{P}} \tag{4}$$

The component of the point cloud representing the height over the ground with respect to the input image shown in Fig. 3a can be seen in Fig. 3b. Then, a pre-segmentation step is applied using knowledge about the application environment in order to remove points outside the application specific and user defined ranges, e.g. points belonging to the ground or too far away and hence error-prone points. In the case shown in Fig. 3a, the codomain for each axis $\mathbb{W}_{x,y,z}$ is

$$\begin{aligned} \mathbb{W}_z &= \{i | (i \in \mathbb{R}) \wedge (0.3 \leq i \leq 4)\}, \\ \mathbb{W}_x &= \{i | (i \in \mathbb{R}) \wedge (8 \leq i \leq 60)\}, \\ \mathbb{W}_y &= \{i | (i \in \mathbb{R}) \wedge (-20 \leq i \leq 20)\} \end{aligned} \tag{5}$$

to remove points which height over the ground is smaller than 0.3 m, higher than 4 m or with a distance to the camera less than 8 m or greater than 60 m. The horizontal interval $\mathbb{W}_y$ has been selected to contain the entire field of view. The estimation of the ranges for $i$ from (5) is shown in section 4.2.




(a) (b) (c)

Figure 3: Presentation of the transformation and filtering process using the portion of the point cloud, which represents the height over the ground in meter. (a) Rectified input image of the left camera, (b) Raw point cloud, (c) Transformed point cloud using the calculated rotation and translation vector with points outside the user defined range removed (marked as dark blue).

The result of the point cloud transformation followed by the point cloud filtering is shown in Fig. 3c where the points representing the ground are removed (marked as dark blue).

This filtered point cloud is further segmented using a Gaussian Mixture Model [SG99], which is an established method for this task. Therefore, the state of a pixel in respect to foreground or background is modelled by several Gaussian distributions which is demonstrated in section 4.3.

As already stated in the introduction, instead of performing the segmentation in 2D space, the component of the point cloud representing the distance to the camera is used as input for the GMM. The reason for this approach is that the calculated point cloud is nearly unaffected by (spontaneous) illumination changes which may occur on construction sites through lights controlled by motion sensors. Additionally, due to the ability of the system to move between specific poses, the illumination of a scene which is currently not in the camera's field of view, can change. This has great impact on GMM training. Since the GMM would not be able to adapt the background model to this change if intensity values were used, this would lead to pixel misclassification. However, this does not occur when using the point cloud for the previously mentioned reason.

The result of the GMM is a binary mask $\mathbf{M}$ with foreground pixels marked as 1 and background pixels as 0. By applying a morphological opening, the noise in the resulting mask is reduced and foreground regions $\mathbf{R}$ can be selected using blob colouring. Nevertheless, there is still a chance that regions are selected falsely, due to the noisy mask. Therefore, small regions are discarded using (6) with $N(a)$ returning the number of pixels and $\rho$ as region size criterion. Using the empirically determined value of $\rho = 0.005$ gives reasonable results. For each region in $\mathbf{R}'$ the corresponding data of the 3D point cloud $\mathbf{P}$ is aggregated, so that we have a set of 3D regions $\mathbf{U}$, see (7).

$$\mathbf{R}' = \{x |\ x \in \mathbf{R}\ \wedge N(x) > \rho \cdot N(\mathbf{M})\} \quad (6)$$

$$\mathbf{U} = \left\{\mathbf{P}(r)\ |\ r \in \mathbf{R}'\right\} \quad (7)$$

However, it turns out that pixels around the marked regions are occasionally selected in error. Hence, those pixels need to be removed to enhance the subsequent estimation of position and size. A common statistical method for outlier elimination is used in which a sorted set of values is divided into four equal sized groups by determining the quartiles $(Q_1, Q_2, Q_3)$ of the set, with $Q_1$ as the median value of the total set, $Q_2$ as the median value of the lower and $Q_3$ of the higher subset. Then, the interquartile distance $I = Q_3 - Q_1$ is calculated which is used to define the so called lower fence $F_L = Q_2 - \eta I$ and upper fence $F_U = Q_3 + \eta I$ with $\eta$ as spreading factor. Finally, values outside the range of $[F_L; F_U]$ are considered as outliers and removed from the dataset.

This method is used to create a set of 3D objects $\mathbf{O}$ by estimating the position $L(\mathbf{U}_i)$ and size $S(\mathbf{U}_i)$ of an object $\mathbf{U}_i$.

$$\mathbf{O} = \{(\mathbf{L}(u), \mathbf{S}(u))\ |\ u \in \mathbf{U}\} \quad (8)$$

The position is then defined by the distances to the optical axis of the virtual left camera in each direction with $L(\mathbf{U}_i) = [d_x, d_y, d_z]$ and the size describes the width and height of the object $S(\mathbf{U}_i) = [w, h]$ whereas $d_z = h$. In case of the distance in the $x$ axis the previously described method is applied to the subset $\mathbf{U}_{i,x}$ which mean value corresponds to $d_x$. All of those values in $\mathbf{U}_{i,x}$ which are marked as outliers are also removed from the set $\mathbf{U}_i$, so that these values are ignored in the following calculations.

The width of the object $\mathbf{U}_i$ is defined as the difference between the left and right edge of the object. These edges are estimated by calculating the mean

of the subsets $\mathbf{M}_l$ and $\mathbf{M}_r$ of $\mathbf{U}_{i,y}$ with $N(\mathbf{M}_{l,r}) \geq 0.1\eta N(\mathbf{U}_{i,y})$, whereas $\mathbf{M}_l$ represents the subset for the left and $\mathbf{M}_r$ for the right edge. Additionally outliers of those subsets are removed using the previously described approach. This could on the one hand lead to inaccurate width estimations in case of high value changes, but on the other hand ensures that the edges are not defined by a single value which might be inaccurate. This procedure is also used to estimate the height $h$ of an object whereas the subset $\mathbf{U}_{i,z}$ is used. Finally, $d_y$ is defined as the left edge of the object increased by half the object's width.

The last step in the processing chain is object tracking. Currently, only a naive method is used to compare the currently detected 3D objects with already known ones. Two objects are considered equal if the size of the newly detected object is in the range of two standard deviations from the already known object's size measurements and the location difference of the detected object to the known one is in the range of two standard deviation from the known object location changes. For that reason, each 3D object contains a list of timestamped size and location measurements.

# 4 EXPERIMENTS

## 4.1 Stereo camera system architecture

To develop and evaluate the method presented in the last section, a dataset of images is required and this in turn requires a stereo camera system. As already mentioned in the introduction, the camera system is motorized to perform pan and tilt movement. This is accomplished by mounting the stereo camera, which consists of two monochrome GigE cameras from TheImagingSource (model DMK-23GM021) with a resolution of $1280 \times 920$ pixel to a pan/tilt system of Invescience LC as illustrated in Fig. 4. The pan/tilt actuators are controlled with a custom application running on a connected PC.

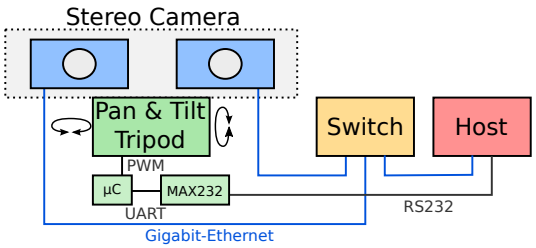



Figure 4: Architecture of the stereo camera surveillance system.

## 4.2 Application model

Designing a stereo camera system for a specific application or environment is a challenging task. This is due to the fact that the system is influenced by various factors, e.g. the baseline width, the image resolution and the elevation of the camera's planned location [LSP+10]. Changing any parameter of the system directly impacts the field of view of the stereo system or the precision of the calculated distance at a pixel [LSP+10]. Furthermore, the range of disparity values also depends on the target application. Additionally, determination of minimum and maximum values of disparity reduces the search space and processing times. Due to this, a model was created which is used to simulate a stereo camera system attached to a wall or pole for a specific surveillance application (see Fig. 5).

With respect to the given application parameters, the theoretically calculated disparity $D$ of an object at a specific distance $Z$ (blue box in Fig. 5) is calculated by (9) and the absolute depth estimation error $|\delta Z|$ is calculated using (10) as stated by Chang and Chatterjee [CC92].

$$D = \frac{bf_x}{Z} \tag{9}$$

$$|\delta Z| = \frac{bf_x}{D^2}\delta D \tag{10}$$

Here, $f_x$ is the focal length in pixels, $b$ is the baseline in meters, $D$ is the disparity and $\delta D$ is the uncertainty of the estimated disparity in pixels, which is assumed to be 1 in the model as a worst case value. By changing the position of the object, the relevant disparity range and the theoretical distance error can be estimated with respect to the stereo system parameters. Additionally, this model is used to estimate the application-specific ranges used in section 3.2 for the pre-segmentation of the point cloud. This model is made publicly available on the Mathworks file exchange platform[2].

## 4.3 Demonstration

Using the estimated parameters of the stereo camera, stereo images for evaluation of the proposed object detection method were acquired. In Fig. 6 the stereo system is shown with a baseline width of 15 cm. The datasets for evaluation were recorded using a baseline width of 55 cm in order to decrease the depth error.

The distance value of a specific pixel is representatively monitored over 592 images (see Fig. 7a and 7b). The pixel's state is modelled using three Gaussian distributions, see Fig. 7c. In Fig. 7b three distinctive situations are shown which correspond to Fig. 7d-7f showing objects crossing that pixel.

[2] `http://mathworks.com/matlabcentral/fileexchange/55420-stereo-camera-application-model`

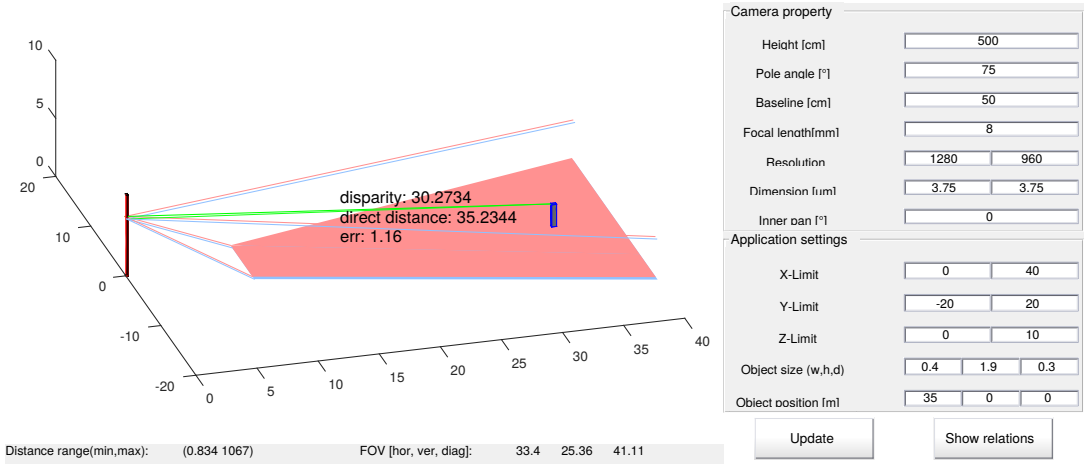



Figure 5: Model of a stereo camera attached to a wall or pole. Areas in blue and red represents the field of view of the left and right cameras. Parameters of the system can be changed with the text fields on the right side. Parameter-dependend informations are shown in the figure.

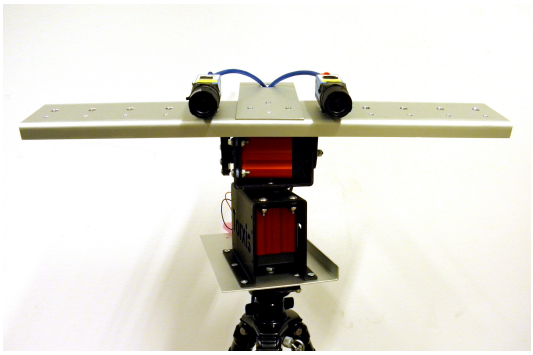


Figure 6: Stereo system with a baseline width of 15 cm and a motorized tripod head for pan and tilt movement.

Table 1 shows the parameters of three distributions over 592 images, sorted by their fitnesses. The dominant distribution is $D_1$ with a mean of 15.686, variance of 0.084 and a fitness of 3.378 representing the background state of the pixel. This demonstrates the feasibility of robust foreground-background segmentation using distance information and a GMM, therefore enabling the performance of object detection using this approach. Additionally, it can be seen that $D_2$ represents the value range of the detected objects. This demonstrates the ability of the GMM to perform multi-modal background modelling.

## 4.4 Evaluation

The method proposed in this paper (in the following referred to as `GMMD`) is evaluated for use in the field of outdoor surveillance, e.g. construction site monitoring, which is influenced by changing illumination and characterized by a dynamic background, by comparing it with other state of the art methods.

| **Parameter** | $\mathbf{D_1}$ | $\mathbf{D_2}$ | $\mathbf{D_3}$ |
|---|---|---|---|
| Variance | 0.084 | 1.673 | 12.352 |
| Mean | 15.686 | 12.904 | 0.778 |
| Weight | 1 | 0.707 | 0.367 |
| Fitness | 3.378 | 0.546 | 0.104 |

Table 1: Parameters of the three distributions $\mathbf{D_1}$, $\mathbf{D_2}$, $\mathbf{D_3}$ modelling the selected pixel's state after 592 images.

A dataset of 3716 timestamped images with a frame rate of 10 images per seconds is recorded. The stereo camera system was placed at the first floor of the Hochschule Emden/Leer covering the campus as already seen in Fig. 3a. This dataset includes situations with global illumination changes, shadow casts and overlapping objects. The first is caused by manual camera aperture manipulation with varying speed to simulate global intensity changes. From theses images, 15 situations are manually labelled using the *Interactive Segmentation Tools* of McGuiness and O'Connor [MO10] for the ground truth data. Thereby, even foreground objects are marked which are already a part of the scene from the beginning of the sequence and are more or less static.

The result of the `GMMD` is compared with the results of the GMM using the greyscale image of the left camera (`GMM`), Frame Differencing (`FD`) and Median Filter (`MD`). The learning rate of the `GMMD` and `GMM` is set to 0.005 because of the image capturing frame rate. Three distributions are used. The initial variances of the `GMMD` and the `GMM` are set to 0.5 and 0.2 respectively. This is due to the fact that the distance estimation is assumed to be more noisy than the grayscale image with values in the range of $[0, 1]$. Additionally, due to the frame rate

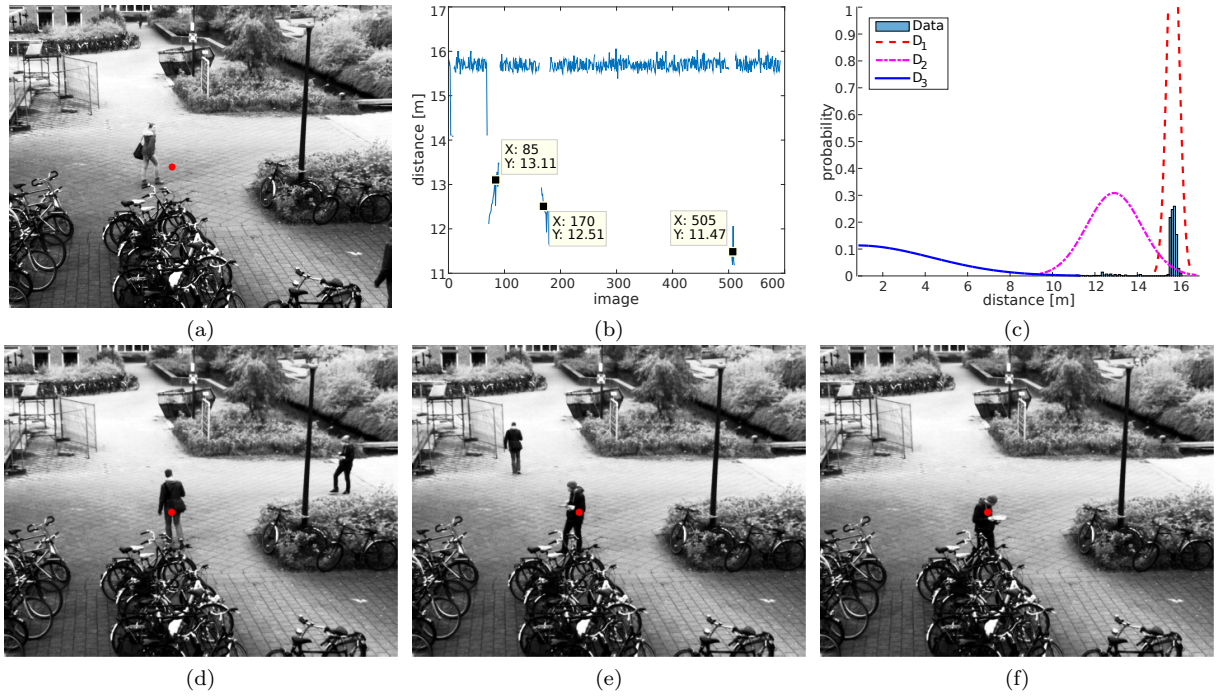



Figure 7: Results of the background modelling process for one pixel over 592 images. (a) Image showing the selected pixel. (b) Plot of the distance values, (c) Plot showing the probability of a distance to occur overlaid with the three distributions of the GMM, (d)-(f) Images of the three distinctive situations.

the FD compares only each fourth image to ensure movement in the image.

For each of the situations and methods, the recall and precision value [SCK04] is calculated to quantify the results of the methods in respect with their resulting foreground masks $M_{\mathrm{GMMD}}, M_{\mathrm{GMM}}, M_{\mathrm{FD}}, M_{\mathrm{MD}}$ and the ground truth mask $G$ using (11) and (12) respectively.

$$\mathrm{Recall}(M) = \frac{|M_{\text{correct marked}}|}{|G_{\text{marked}}|} \tag{11}$$

$$\mathrm{Precision}(M) = \frac{|M_{\text{correct marked}}|}{|M_{\text{marked}}|} \tag{12}$$

The results of the methods are shown in Fig. 8 whereas $T$ is a threshold for classifying foreground and background pixel with respect to the pixel value changes. In general, methods performing well have a high recall and precision value. However, it is evident that none of the tested methods reach a recall value greater than 62%, which in some extend depends on the ground truth masks. This is due to the fact that static objects are also labelled as foreground but can not be detected by the tested methods. The results show the GMMD performing reasonable well in all situations with an average precision of 72.8% and a relatively low standard deviation of 8.5%. No other method has shown behavior this robust (see table 2). For instance, the precision of the GMM has a much higher standard

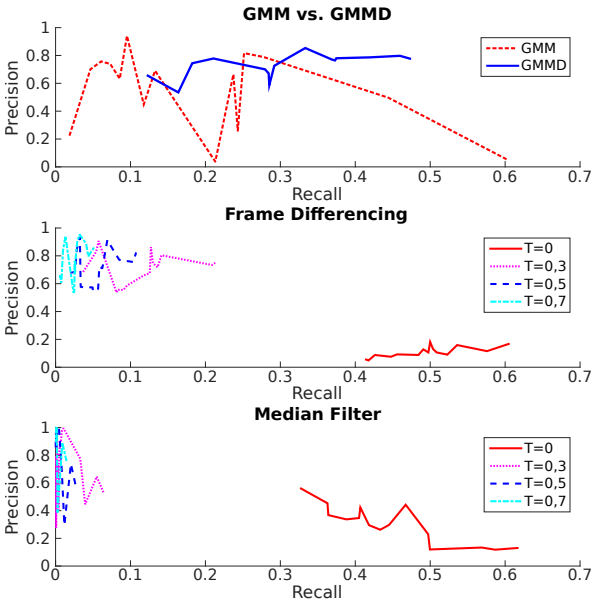



Figure 8: Results for the manually labelled images

deviation (28.4%). In Fig. 9 five prominent situations are shown which describe the behaviour of the GMM. The situation (3) corresponds to the rightmost dataset of Fig. 8 with a precision of 5.24% because of a sudden change in the global intensity which results in an inverted foreground mask [SCK04]. The proposed GMMD however is unaffected by these changes and produces a foreground mask with a precision of 77% and a recall of 37%. The situations (2) and (5) in Fig. 9 show the classification

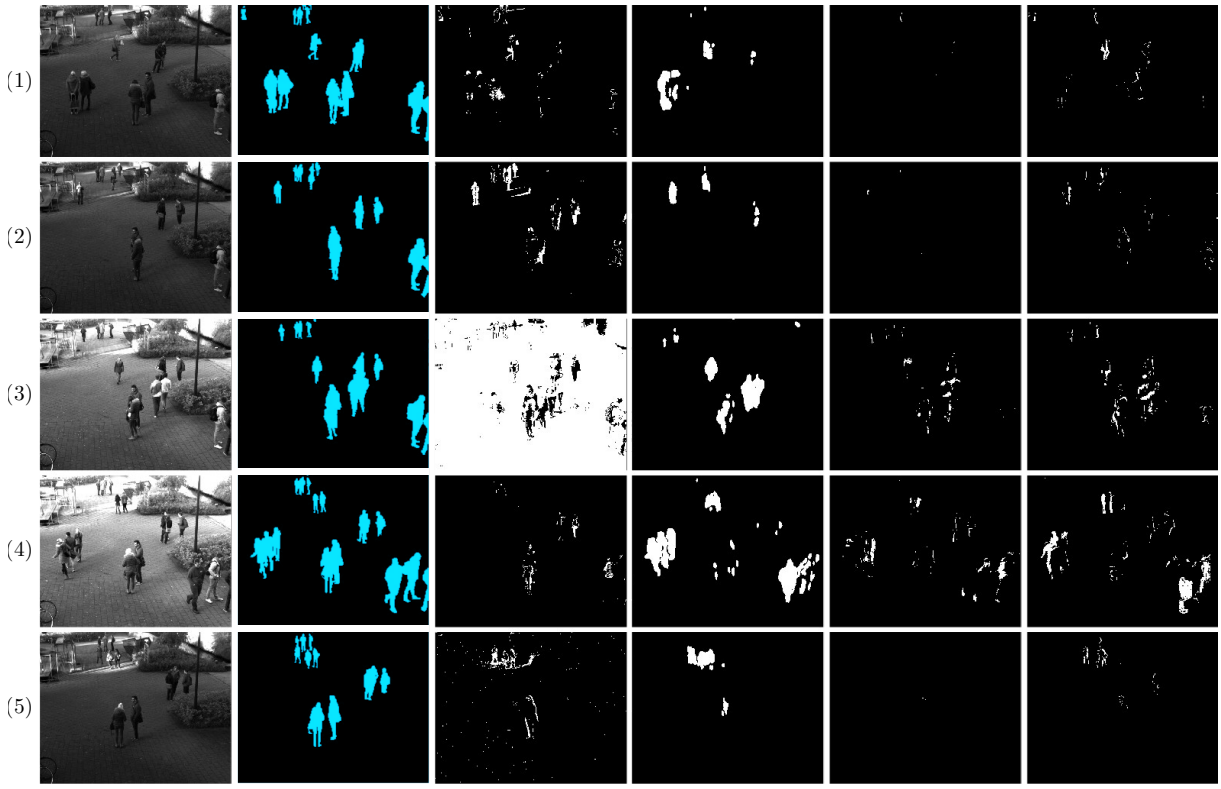



Figure 9: Results of the methods for five prominent situations. (a) Rectified left image, (b) Ground truth foreground mask, (c) Result `GMM`, (d) Result `GMMD`, (e) Result `MF`, (f) Result `FD`

of shadow casts as foreground pixels in the case of the `GMM`. However, situation (1) and (2) shows an advantage of the `GMM` over the `GMMD` because it even detects far-away objects.

| | Recall | | Precision | |
|---|---|---|---|---|
| **Dataset** | **Mean** | **Std. Dev.** | **Mean** | **Std. Dev.** |
| GMM | 0.194 | 0.164 | 0.551 | 0.284 |
| GMMD | 0.308 | 0.106 | 0.728 | 0.085 |
| MF ($T=0$) | 0.452 | 0.086 | 0.301 | 0.137 |
| MF ($T>0$) | 0.008 | 0.014 | 0.798 | 0.220 |
| FD ($T=0$) | 0.492 | 0.055 | 0.109 | 0.039 |
| FD ($T>0$) | 0.065 | 0.050 | 0.732 | 0.127 |

Table 2: Results of the experiment. The results of `MF` and `FD` using a threshold greater zero are merged.

## 5 CONCLUSION

In this paper an approach for object detection in the field of outdoor surveillance for e.g construction site monitoring was presented which combines an actuated stereo camera system with camera pose specific Gaussian Mixture Models (GMM). A novel processing chain for detection of objects based on a calculated 3D point cloud and the current camera pose was described. Additionally, the actuated stereo surveillance system is described and a model is presented for the estimation of application-specific parameters which simplifies the stereo camera system design process.

Furthermore, the presented approach is compared to other state of the art methods using a self captured image database. With an average precision of 72.84% and a recall value of 30.82% it outperforms the other methods. Additionally, it was shown that the proposed method is robust against changing illumination and shadow casts which often occurs in outdoor surveillance applications like construction site monitoring even while moving the stereo camera. However, overexposed pixels cause an incomplete distance map due to the pixel correlation process for the disparity calculation and hence lead to an inaccurate segmentation and the detection range of the method is limited by the stereo camera's distance calculation error.

For a more robust identification of objects, the current naive matching and tracking method need to be extended in future work. Additionally, for $1280 \times 960$ images, the current disparity map calculation time on a single threaded i7-3770 PC is 0.44 s per image without hardware acceleration. Follow-up works on GPU, FPGA [GEM09] or even on CPU [SLAR14] should lead to a higher number of frames

analysed per second. Additional performance gains can be achieved by parallelization of the processing chain presented in section 3.2 using a pipeline architecture which is typically used in the field of processor design.

Currently, the ground is assumed to be plane which is typically not the case in the real world. Hence, the ground selection phase of the online calibration process need to be extended to build a more realistic model of the ground in order to improve the object measurement and localization.

Furthermore, the application of the morphological operation for noise reduction could split up objects into multiple regions and hence multiple 3D objects which degrades the object detection rate. In follow-up works this can be addressed by clustering regions with respect to their 3D representation to enhance the object detection process.

## REFERENCES


[CC92] C. Chang and S. Chatterjee. Quantization Error Analysis in Stereo Vision. In *Signals, Systems and Computers, 1992. 1992 Conference Record of The Twenty-Sixth Asilomar Conference on*, pages 1037–1041 vol.2, October 1992.

[Cor11] Peter I. Corke. *Robotics, Vision & Control: Fundamental Algorithms in Matlab*. Springer, 2011.

[DB04] J. Douret and R. Benosman. A multi-cameras 3D volumetric method for outdoor scenes: a road traffic monitoring application. In *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference*, volume 3, pages 334–337 Vol.3, Aug 2004.

[FR97] Nir Friedman and Stuart Russell. Image Segmentation in Video Sequences: A Probabilistic Approach. In *Proceedings of the Thirteenth conference on Uncertainty in artificial intelligence*, pages 175–181. Morgan Kaufmann Publishers Inc., 1997.

[GEM09] Stefan K. Gehrig, Felix Eberli, and Thomas Meyer. A Real-Time Low-Power Stereo Vision Engine Using Semi-Global Matching. In Mario Fritz, Bernt Schiele, and Justus Piater, editors, *Proceedings of the 7th International Conference on Computer Vision Systems: Computer Vision Systems*, volume 5815 of *Lecture Notes in Computer Science*, pages 134–143. Springer, 2009.

[HHD98] Ismail Haritaoglu, David Harwood, and Larry S. Davis. $W^4S$: A Real-Time System for Detecting and Tracking People in 2 1/2D. In *Computer Vision-ECCV'98*, pages 877–892. Springer, 1998.

[KMP09] Sanjeev Kumar, Christian Micheloni, and Claudio Piciarelli. Stereo Localization Using Dual PTZ Cameras. In *Computer Analysis of Images and Patterns*, volume 5702 of *Lecture Notes in Computer Science*, pages 1061–1069. Springer Berlin Heidelberg, 2009.

[Kov00] P. D. Kovesi. MATLAB and Octave Functions for Computer Vision and Image Processing, 2000. Available from: `http://www.peterkovesi.com/matlabfns/`.

[LCK13] Honghai Liu, Shengyong Chen, and Naoyuki Kubota. Intelligent Video Systems and Analytics: A Survey. *Industrial Informatics, on IEEE Transactions*, 9(3):1222–1233, August 2013.

[LSP+10] David F. Llorca, Miguel A. Sotelo, Ignacio Parra, Manuel Ocaña, and Luis M. Bergasa. Error Analysis in a Stereo Vision-Based Pedestrian Detection Sensor for Collision Avoidance Applications. *Sensors*, 10(4):3741–3758, 2010.

[MO10] Kevin McGuinness and Noel E. O'Connor. A comparative evaluation of interactive segmentation algorithms. *Pattern Recognition*, 43(2):434–444, February 2010.

[SCK04] S. Cheung Sen-Ching and Chandrika Kamath. Robust techniques for background subtraction in urban traffic video. In *Electronic Imaging 2004*, pages 881–892. International Society for Optics and Photonics, 2004.

[SG99] Chris Stauffer and W.E.L. Grimson. Adaptive background mixture models for real-time tracking. In *Computer Vision and Pattern Recognition, on 1999. IEEE Computer Society Conference*, volume 2. IEEE, 1999.

[SLAR14] R. Spangenberg, T. Langner, S. Adfeldt, and R. Rojas. Large scale semi-global matching on the cpu. In *Intelligent Vehicles Symposium Proceedings, 2014 IEEE*, pages 195–201, June 2014.

[ZOS13] Guang Zheng, Shunichiro Oe, and Zengguo Sun. Moving Object Tracking and 3D Measurement Using Two PTZ Cameras. In *Intelligent Networking and Collaborative Systems (INCoS), on 2013 5th International Conference*, 2013.

[ZWW10] Jie Zhou, Dingrui Wan, and Ying Wu. The Chameleon-Like Vision System. *Signal Processing Magazine, IEEE*, 27(5):91–101, September 2010.

# Fisheye Lens Correction by Estimating 3D Location

Ji-Won Lee
Department of Electronics Engineering,
Ewha Womans University
52 Ewhayeodae-gil, Seodaemun-gu,
Seoul 03760, Korea
leejw9991@ewhain.net

Byung-Uk Lee
Department of Electronics Engineering,
Ewha Womans University
52 Ewhayeodae-gil, Seodaemun-gu,
Seoul 03760, Korea
bulee@ewha.ac.kr

## ABSTRACT

Since a fisheye lens can capture a wide angle scenery, it is broadly used for surveillance or outdoor sports. However, acquired images suffer from severe geometric distortions. Most of the existing distortion correction algorithms depend on linear features: images of linear features are first identified and then 2 dimensional warping is applied to make the curved images look straight. We propose a novel fisheye distortion correction method that estimates 3 dimensional (3D) locations of a foreground first, and then projects them to an image plane by perspective projection. When we know approximate distance of the foreground object, as in cases of head mounted camera, we can assume the 3D object plane of the foreground, and then estimate the 3D location from image points after internal camera calibration. For head mounted camera, foreground is a face and body of a human, and distortion of human figure is quite unnatural and awkward. Moreover, human figures lack linear features which excludes the use of conventional 2D warping techniques. We present techniques to estimate the 3D position from a corresponding 2D image point, which enables calculation of 3D object location. And then apply perspective projection to the 3D object position to obtain a distortion-free image. We demonstrate the efficacy of the proposed method using fisheye camera images and the applicability of the proposed concept to real applications.

## Keywords

Fisheye Lens, Lens Distortion Correction, Radial Distortion

## 1. INTRODUCTION

Fisheye lens can capture extremely wide view, therefore, it is useful for surveillance or outdoor sports camera. However, fisheye images have geometric distortion which is more severe toward the boundaries of the image. Moreover, depending on the position, foreground objects or a human is distorted severely, and then the fisheye image looks unrealistic. Therefore it is necessary to correct for the geometric distortion to restore natural and realistic figures.

There are several methods on correcting fisheye lens distortion by processing on viewing spheres [Sha10a], [Cha13a], [Car09a], [Wei12a], [Kan06a]. Sharpless [Sha10a] and Chang [Cha13a] introduced two-step projection methods from viewing sphere which can map from wide angle images into image planes. Sharpless firstly maps the sphere image into equirectangular space and then rectilinear-projects by adjusting a scale controlling a distance between the center of the projection and the view plane. He applied the scale into azimuth angle and altitude angle and then obtained corrected image coordinates. It makes the regions to be horizontally compressed that are between the radial lines from the central vanishing point. This concept was based on the paintings of Gian Paolo Pannini and many painters of Italian Baroque period: they created the wide angle paintings using standard perspective projection, but the distortion of the projection was not shown in the paintings. Sharpless [Sha10a] adopted the method to correct for the distorted wide angle scenery. Chang [Cha13a] further developed Sharpless' two step projection method from the viewing sphere to an image plane in [Cha13a]. For an initial projection, Chang [Cha13a] introduced a swung surface which was created by finding parameters linearizing line segments, which were found in 6 faces obtained by box projection. After the initial projection to the surface, the surface was projected to image plane using perspective projection. Then, this method conserved the linearity of horizontal lines better than those of vertical direction. While two methods established projection models, Carroll [Car09a] and Wei [Wei12a] adaptively corrected for the fisheye lens distortion by using user inputs, such as line constraints that should be conserved as straight lines. Carroll [Car09a] used line constrains users entered and dealt with the straightness of them, neighbor pixels' conformality and smoothness to warp each pixel in order to obtain natural images having little fisheye distortion. In other

words, for each pixel, they find a new image coordinates optimizing sum of several energies such as straightness, conformality and smoothness. In [Car09a], Carroll used static images, while Wei [Wei12a] utilizes video images of several frames. Wei [Wei12a] carried out the above process tracing input line constraints over consecutive frames. However, these methods could not work properly when the number of inputs were inadequate or the fisheye images were severely distorted.

We propose a new fisheye lens distortion correction method using 3D position information with a prior on the distance, whereas current methods focus on conserving the linearity of straight lines, and mapping in 2D. In a situation of capturing an outdoor activity by a head mounted camera, the distance between the human and the camera is almost fixed, thus we can use the distance between them for distortion correction. If the foreground is assumed to be on a plane of known distance, we can calculate the 3D position of the foreground from images coordinates after inverse projection. Then, we can project these 3D coordinates using perspective projection, and reconstruct the foreground image without nonlinear geometric distortion. Usually, the foreground and camera is close and the solid angle covered by the object is considerable, which results in severe image distortion. Substantial distortion of human face or body is quite unpleasing, however, because there is no straight line features in human images, it is not possible to apply the existing distortion correction methods. The proposed algorithm does not apply 2D image warping; the novel method estimate the 3D position of the foreground object from a prior and the image position, and then apply perspective projection to the estimated 3D location for generating distortion corrected images.

In order to correct for the distortion of foreground in fisheye lens images, we firstly segment the foreground in captured images, estimate 3D coordinates of them assuming the distance to the foreground, and then project 3D coordinates into the image coordinates by perspective projection where the mapped coordinate is inversely proportional to the distance to the lens.

The paper is organized as follows. Section 2 describes the proposed method which is estimating 3D location of foreground and then perspective projection of the foreground to an image plane. Section 3 shows the experimental results. Lastly, Section 4 presents conclusions and future work.

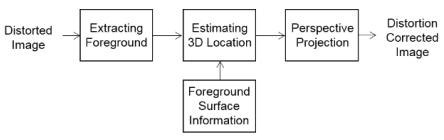



**Figure 1. Block diagram of the proposed method**

## 2. PROPOSED METHOD

We describe an innovative concept of distortion correction: virtual 3D coordinates of foreground objects are estimated first and then re-project them to obtain distortion-free images.

Figure 1 shows a block diagram of the proposed process. As shown in Figure 1, we initially segment the foreground object from an image, and then estimate 3D locations of the foreground using image coordinates assuming planar object plane. The distance and the orientation of the plane is assumed to be known. For head mounted cameras, the distance from the camera to the foreground is stable and the image distortion is not sensitive to perturbation of the distance, therefore, distortion correction can be accomplished.

Once 3D coordinates are estimated, geometric distortion can be completely eliminated without linear features. We can apply perspective projection to generate an image without nonlinear distortion.

### 2.1 Internal Calibration

Fisheye camera lens model can be characterized as a function of image distance $\boldsymbol{r}$ and an angle $\boldsymbol{\theta}$, where $\boldsymbol{r}$ is the distance between the principal point and the image position $p$ (in Figure 2(b)) on the image plane, and $\boldsymbol{\theta}$ represents the angle between the incoming ray from the 3 D point $P$ (in Figure 2(b)) and the optical axis of the camera. Figure 2(a) shows curves of several fisheye lens models and Figure 2(b) represents an image plane $(x, y)$ and the camera lens coordinates in 3D space $(X_c, Y_c, Z_c)$, which represents the lens model.

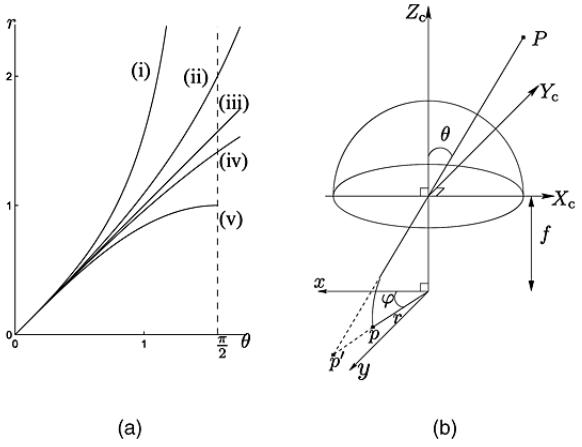



**Figure 2. (a) Various models of fisheye lens. Lens models can be represented as $r - \theta$ curve with focal length $f = 1$ where $r$ is the distance from the principal point to a point $p$ in the image plane and $\theta$ is an azimuth angle of object position $P$. (b) Fish-eye camera model setup. The image point of the point $P$ is $p$ whereas it would be $p'$ by a pinhole camera [Kan06a].**

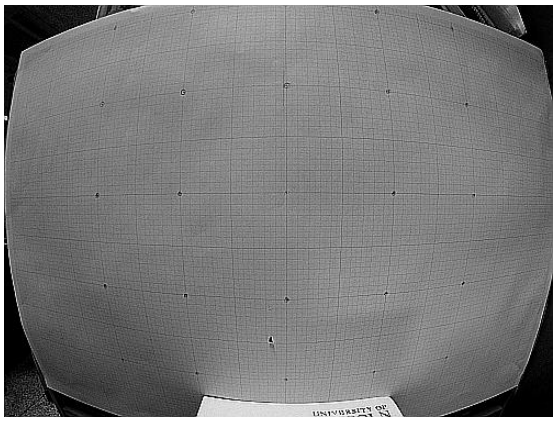


**Figure 3. Fisheye image of a grid pattern for internal calibration captured by GoPro HERO4 Silver. Optimized focal length and principal point are found by using the coordinates $(u, v)$ of this image.**

We used GoPro HERO4 Silver camera model and we found that the orthographic model ((v) in Figure 2(a)) fits the distortion of the camera. In Figure 4, $\theta - r$ curves from experiment (red dot) and by orthographic lens model (blue line) are plotted together, which shows good agreement of the observed data and the orthographic model. The proposed method can be applied to any lens distortion model.

By finding an optimized focal length and principal point for this model, we enhance the accuracy for estimating the 3D coordinates. For orthographic lens, projection can be described by the following equation,

$$r = f \sin\theta, \qquad (1)$$

where $r$ is the image distance from the principal point $(u_0, v_0)$ to an image point $(u, v)$. We used a grid pattern perpendicular to the optical axis as shown in Figure 3. Then the tangent of angle $\theta$ is $\frac{\ell}{d}$ where $d$ is the distance from the optical center to the center of the grid pattern $(x_0, y_0)$ and $\ell$ is the distance from $(x_0, y_0)$ to the grid point $(x_p, y_p)$. The image positon of $(x_0, y_0)$ corresponds to the principal point. We applied affine transform to establish a mapping of the center of the grid pattern and the principal point [Lee16a]. To find the optimized internal parameters, focal length and principal point, we find the minimum mean square error solution:

$$\arg\min_{f,u_0,v_0} \sum_i \frac{(r_i - f\sin\theta_i)^2}{N}, \qquad (2)$$

where N is the number of data points.

## 2.2 Estimating 3D Location of Foreground

Once we have an equation of the foreground plane in 3 D space, then we can find the 3D location of a point from a calibrated image point. The intersection of the optical axis and the foreground plane is (0, 0, $d$) and the tilt angle is $\alpha$; the rotation axis of tilt is the

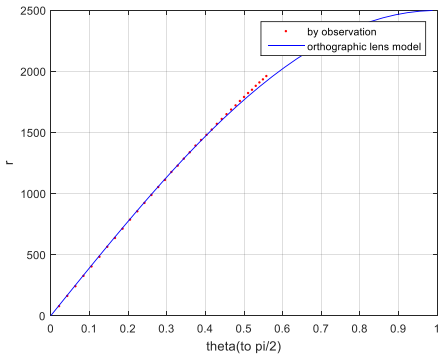



**Figure 4. Observed $\theta - r$ curve (red dot) is obtained by calculating $\theta$ and $r$ using 36 image points which are 1cm apart horizontally from the principal point of the image (max $\theta = 56.2^\circ$). Theoretical $\theta - r$ curve (blue solid line) is drawn from the orthographic lens model.**

$y$-axis (in Figure 5) for simplicity. We derive the 3Dlocations of the foreground by finding the intersection of the image ray and the object plane.

Figure 5 shows a 3D space with a surface and $x$, $y$, $z$--axis. Also the center of the lens is shown as the origin $\boldsymbol{O}$. As shown in this figure, we can derive the coordinates on the object surface $(x_p, y_p)$ when we know image coordinates $(u, v)$.

Figure 5 also illustrates the relationship of the object surface coordinates and the image coordinates. The principal point $\boldsymbol{O'}$ is $(u_0, v_0)$, and the angle between a ray from the object center to a surface point P and a horizontal axis of the surface is $\varphi$ which is obtained from the following equation,

$$\varphi = \tan^{-1}\left(\frac{u - u_0}{(v - v_0)/\cos\alpha}\right), \qquad (3)$$

$u$ and $v$ are the image coordinates along the $x$-axis and $y$-axis of Figure 5, respectively. Since the foreground surface $S$ is tilted by the angle α along the axis of $y$, we divide $v - v_0$ by $\cos\alpha$ to compensate for the foreshortening. The direction angle $\varphi$ on the foreground surface can be obtained from the image coordinates because we know the tilt angle of the foreground surface. The vertical axis of the surface is the same as that of the image plane, whereas the horizontal axis is rotated by α. And then, we can calculate the image distance $r$,

$$r = \sqrt{(u - u_0)^2 + (v - v_0)^2}, \qquad (4)$$

which is the Euclidean distance from the principal point $(u_0, v_0)$ to an image point. Moreover, we can obtain the angle $\theta$, in the 3D space, using the calibrated lens model of the equation 1.

With the known angles $\varphi$, and $\theta$, we are ready to calculate the distance $\ell$ which is the distance from the center of the object surface to a point on the

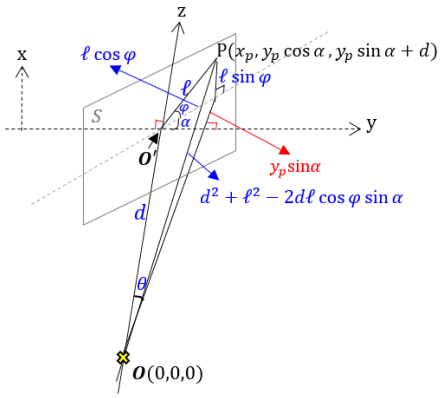



**Figure 5. $(x_p, y_p)$ of the foreground surface $S$ can be derived using the 3D relations in terms of the known priors, such as the distance $d$ between the center of the lens $O$ and the center of the surface $O'$ and tilt angle of the surface $\alpha$, and parameters $\theta$ and $\varphi$ derived from image coordinates. Also the perspective projection is the mapping of each 3D coordinate to the image position depending on the value $z$. The depth $z$ is calculated from $\ell$ as in the figure because we know $\varphi$ and $\alpha$ already from the image.**

surface. The equation for $\ell$ can be derived as,

$$\ell = \frac{-d\sin\theta(w\sin\theta - \sqrt{1-w^2}\cos\theta)}{w^2 - \cos^2\theta}, w = \cos\varphi\sin\alpha\,. \quad (5)$$

Since the image direction angle $\varphi$ is obtained by equation 3, we could calculate the object image location $(x_p, y_p)$ on the surface plane by the following formula,

$$(x_p, y_p) = (\ell\cos\varphi\,, \ell\sin\varphi) \quad (6)$$

In other words, the surface coordinates are reconstructed from the image coordinates and object surface information through equations 3 to 6.

## 2.3 Rendering an Image without Distortion

In order to render an image without distortion, we apply perspective projection to the surface coordinates $(x_p, y_p)$. And then, perspective projected foreground is overlaid on the input image so that we can obtain a distortion corrected foreground over a wide angle background image.

Firstly, we need to find the 3D coordinates of the object surface points so that we can apply perspective mapping. The 3D coordinates $(x, y, z)$ are derived by the following equation:

$$x = x_p \quad (7)$$

$$y = y_p\cos\alpha \quad (8)$$

$$z = y_p\sin\alpha + d, \quad (9)$$

the relation between an object image position $(x_p, y_p)$ which is the surface coordinates and the 3D camera coordinates $(x, y, z)$ are shown in Figure 5.

Once we obtain the 3D position, we can calculate image coordinates which are finally mapped to the corrected image. At this stage, we adjust the range of $z$ values because it controls the ratio of the sizes of the near and far foregrounds.

We refer the parameter to adjust the range of $z$ as $zRangeScale$, and the scaled $z$ values are represented with $z_s$ as in the following equation,

$$z_s = (z - m_z)zRangeScale + m_z, \quad (10)$$

where $m_z$ is the minimum $z$ value of the foreground and $zRangeScale$ is the z-length of the foreground object.

After finding $z_s$ for each $(x_p, y_p)$, we derive a corresponding image coordinates $(u, v)$ which is the result of perspective projection by the following formula,

$$u = g_s(m_z / z_s)x + u_0 \quad (11)$$

$$v = g_s(m_z / z_s)y + v_0\,, \quad (12)$$

where $g_s$ controls the global size of the foreground object on the resulting image plane. The principal point $u_0$ and $v_0$ are added to shift the (0, 0) principal point of the camera model to the actual principal point on the image coordinates.

# 3. EXPERIMENTS

After finishing internal camera calibration which estimates internal parameters of the camera, such as the focal length and the principal point, we estimate the 3D coordinates of an object from image coordinates. In this section, we verify the accuracy of each step and analyze experimental results.

## 3.1 Estimation of 3D Coordinates

We applied the calibration process which is described in Section 2.1 by using an image of a tilted grid pattern. This tilted grid pattern image has distortions, therefore the same interval is shown differently depending on the position in the image as in Figure 6. However because the grid pattern shows the ground truth location, we can find the error of the estimated position easily. We verified that the derived equations for calculating the 3D object plane coordinates $(x_p, y_p)$ are accurate with the test results. In this experiment, we optimize the distance $d$ and the angle $\theta$ by using 30 image coordinates of horizontal points on the tilted grid image of Figure 6.

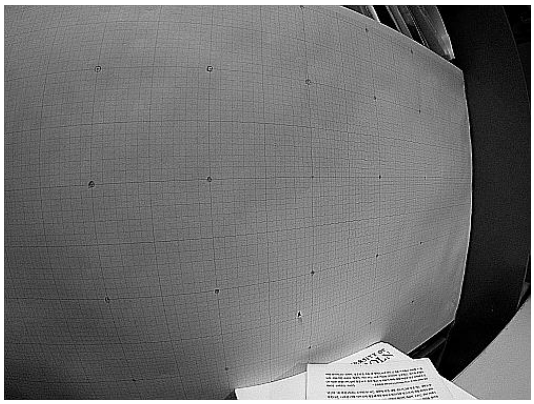


**Figure 6. A fisheye image of a tilted plane captured by GoPro HERO4 Silver. The plane is set up away from the lens by 30cm, and rotated horizontally by $30^{\circ}$ ($d \approx 30\text{cm}, \alpha \approx 30^{\circ}$).**

We can verify the accuracy of the calculated coordinates $(x_p, y_p)$ because we know the true coordinates by the reading the graph and then compare with the calculated position. We represent a result showing both the true and obtained $\ell$ which is the distance from the center of the foreground surface, and calculate an error between them as in Figure 7. We use Figure 6 for this task. It is an image of a grid pattern 30 cm away and with $30^{\circ}$ tilt.

In Figure 6, horizontal 30 points of 1 cm interval are used for calculating foreground image position $(x_p, y_p)$. The distances $\ell$ between the center of the grid paper (0,0) and the coordinates $(x_p, y_p)$ are shown with the corresponding ground truth coordinates in Figure 7. Also we calculate the maximum absolute error (MAE) and mean square error (MSE) for the 30 data points after optimizing the $d$ and $\alpha$. The errors show quite accurate results: the root mean square error of 30 points is 0.1 cm and the maximum error is 0.24 cm. Therefore the defined equations are reliable for calculating the 3D coordinates. Based on the fact that the derived equations can estimate the 3D positions with high reliability, we can reconstruct the 3D foreground surfaces from image coordinates of real scenes in the following subsection with high accuracy.

## 3.2 Perspective Projection Method

We apply a perspective projection for re-projecting the obtained 3D coordinates to the image plane to correct for the fisheye lens distortion of the foreground image. We use indoor/outdoor real images for finding 3D coordinates of foregrounds, and then reconstruct a distortion-free result image by the perspective-projection method of section 2.3.

We correct for the geometric distortion of a human figure in Figure 8(a). Firstly, a foreground object is segmented using MATLAB toolbox Image Segmenter.

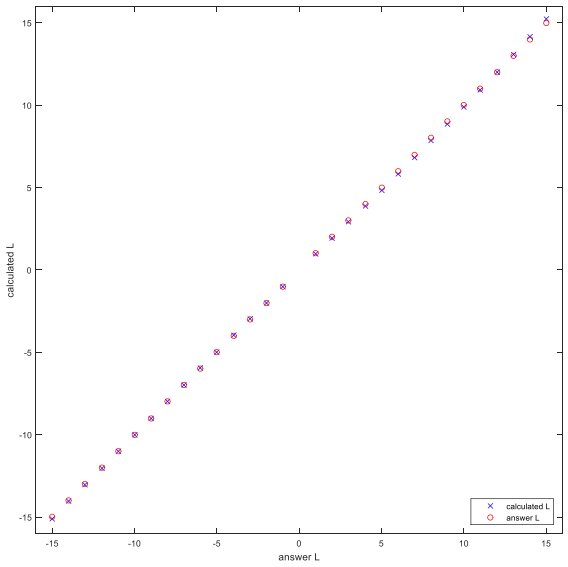



**Figure 7. Calculated (blue ×) and ground truth (red ○) distance $\ell$. $\ell$ is calculated by the proposed equations using the foreground information of the distance, the tilt angle, and the image coordinates.**

And then if we know the distance from the lens to the 3D foreground surface and the tilt angle of the foreground, then we can derive 3D coordinates of every foreground pixel from the extracted image coordinates and the prior. The reconstructed object surface from the original image is represented in Figure 8(b). The coordinates of the object plane is $(x_p, y_p)$ as explained in Section 2.2. The recovered foreground is perspective projected and overlaid to the original image as shown in Figure 8(c). We can notice that the human figure looks more natural after the distortion correction using the proposed method. It is hard to find linear features in this human figure image, therefore, previous distortion correction cannot be applied to this class of images. We apply the correction method to Figure 9(a) and the resulting image, Figure 9(c), shows more familiar looking human figure with better proportions. The background of this image is window frames and floor, which can be approximated as two separate planes in 3D space. We applied the same correction algorithm to the background, and the result image is Figure 10. Input image Figure 9(a) is divided into the figure foreground, the floor background and the window frame background, and then 3D position of each object plane is recovered and then perspective projected separately. This result shows excellent correction of distortion of the human figure and the background. Merging of the foreground and background is not perfect yet, therefore, we notice mismatch between them. The performance of the proposed distortion correction algorithm heavily depends on segmentation and merging of the foreground.

(a) (b) (c)

**Figure 8. (a) Original Image, (b) recovered foreground image, (c) overlay of the perspective projection of the foreground**

(a) (b) (c)

**Figure 9. (a) Original Image, (b) recovered foreground image, (c) overlay of the perspective projection of the foreground**

## 4. CONCLUSION

Most of previous researches corrected fisheye lens distortion by finding features of straight lines and then applied 2D warping to make the images of linear features straight. We propose a novel correction method for fisheye lens distortion. For cases where the distance between the lens and the foreground object/human is stable, for example head mounted cameras, we propose an algorithm to correct for the fisheye distortion by estimating 3D locations of the foreground. In order to fulfill this objective, we derived techniques for estimating 3D positions from image coordinates of the foreground and apply a perspective mapping to the estimated 3D coordinates to render a distortion-free image of the foreground. Future work includes application of the proposed method when the 3D foreground is composed of many planes or curved surfaces. Estimation of the depth or distance to the foreground is also an interesting topic.

**Figure 10. Perspective projection of the background (the window frames and the floor) with the overlay of the perspective projection of the foreground human figure.**

## 5. REFERENCES

[Car09a] Carroll, R., Agrawala, M., & Agarwala, A. Optimizing content-preserving projections for wide-angle images. ACM Transactions on Graphics, **28**(3), Article No. 43, 2009.

[Cha13a] Chang, C. H., Hu, M. C., Cheng, W. H., & Chuang, Y. Y. Rectangling Stereographic Projection for Wide-Angle Image Visualization. In Proceedings of the IEEE International Conference on Computer Vision, pp. 2824-2831, 2013.

[Kan06a] Kannala, J., & Brandt, S. S. A generic camera model and calibration method for conventional, wide-angle, and fish-eye lenses. IEEE Transactions on Pattern Analysis and Machine Intelligence, **28**(8): 1335-1340, 2006.

[Lee16a] Lee J., Lee B. Optimized modeling of orthographic fisheye lens by estimating principal point. In Proceedings of Image Processing and Image Understanding Workshop, Vol. 28, pp. 727-729, 2016.

[Sha10a] Sharpless, T. K., Postle, B., & German, D. M. Pannini: a new projection for rendering wide angle perspective images. In Proceedings of the Sixth international conference on Computational Aesthetics in Graphics, Visualization and Imaging. Eurographics Association, pp. 9-16, 2010.

[Wei12a] Wei, J., Li, C. F., Hu, S. M., Martin, R. R., & Tai, C. L. Fisheye video correction. IEEE Transactions on Visualization and Computer Graphics, **18**(10): 1771-1783, 2012.

# Comparative Visual Analysis of Transport Variability in Flow Ensembles

Mihaela Jarema
TU München
Boltzmannstrasse 3
85748, Garching bei München, Germany
mihaela.jarema@tum.de

Johannes Kehrer
TU München
Boltzmannstrasse 3
85748, Garching bei München, Germany
johannes.kehrer@tum.de

Rüdiger Westermann
TU München
Boltzmannstrasse 3
85748, Garching bei München, Germany
westermann@tum.de

## ABSTRACT

We propose a novel approach that enables a comparative visual exploration of the transport variability in ensembles of 2D flow fields. To reveal when and where divergences in transport occur, we first present a new approach to analyze the time-varying pairwise dissimilarities of ensemble trajectories, by using Gaussian Mixture Models (GMMs) to identify the distribution modes and the Mahalanobis distance to refine the dissimilarity measures. This enables drawing enhanced spaghetti plots, by using the color of the contour of each trajectory to encode the temporal evolution of the member, and the opacity for its representativeness relative to the ensemble behavior. To also allow a global view of the transport variability across selected sub-domains, we introduce a new graphical abstraction based on the visualization of miniaturized versions of the enhanced spaghetti plots in a small-multiples layout. To achieve this, we propose a new kind of downscaling that preserves the relevant trends in the transport behavior. We have designed a user interface comprising multiple linked views to visualize simultaneously global and local transport variations, as well as how similar the transport behavior of the ensemble members is.

## Keywords

Uncertainty Visualization, Ensemble Vector Field Data, Time-varying Data.

## 1 INTRODUCTION

In many scientific disciplines, ensemble simulations are used to estimate the uncertainty inherent in the development of physical fields, by providing representative samples of the possible states that could evolve out of perturbed initial conditions and different models. Weather forecasting is one such example, where multiple forecasts are performed simultaneously to obtain probabilities of occurrence of specific weather events. Analyzing the temporal variability of an ensemble helps determine when and where divergences occur, and, implicitly, the locations where and the time intervals over which a simulation is more or less reliable.

Analyzing the ensemble variability is, however, not straightforward: Firstly, transport divergences occur gradually over time, due to spatial variations of the transport paths and the transport velocity along them. Analyzing the flow variability requires new concepts to determine these divergences and, in particular, when the divergences started occurring. Secondly, visualizing the temporal variability of an ensemble – locally, at a selected domain point, or globally, to compare the transport across the domain – is challenging, since it requires new graphical abstractions to show the complex spatio-temporal ensemble evolution in an intuitive way.

**Contribution:** We propose a visual analytics approach to address the aforementioned challenges and explore the temporal variability of ensembles of vector fields. We provide novel means to perform a local and global visual analysis of the dissimilarities of ensemble members across the domain.

To analyze the transport deviations over time statistically, we introduce the application of the Mahalanobis distance on Gaussian Mixture Models (GMMs). Approximating the distributions of tracer particles at every grid point with mixtures of Gaussian modes enables the use of the Mahalanobis distance to assess the pairwise member dissimilarities relative to the variability allowed by each mode. To quantify the ensemble divergence over time further, we define the divergence count of an ensemble member per time step as the (normalized) number of members dissimilar to it. As members evolve, their pairwise dissimilarities and divergence counts change, revealing the time steps and locations of new (dis)similarities to others members.

The information we derive from the pairwise dissimilarities and divergence counts is then encoded graphically, to enable an effective local and global visual analysis. We propose the following novel approaches:

- **Divergence Visualization:** We enhance the spaghetti plots of the ensemble trajectories by encoding the transport evolution and divergence counts over time; this shows when and where ensemble members behave (dis)similarly, and is especially effective in cases when trajectories exhibit similar geometric shapes, but trajectory points considered at the same time instant are dispersed.
- **Small-Multiples:** We compare the transport variability across selected sub-domains via a new graphical abstraction based on miniaturized versions of enhanced spaghetti plots in a small-multiples layout. We present a new downscaling method that preserves the relevant trends in the transport behavior.
- **Similarity visualization:** We cluster ensemble members based on their flow similarity across the domain and visualize the dynamical evolution of the clusters using parallel sets.

The proposed approaches are combined into an interactive visual analytics tool, to give insight into the various aspects of the flow variability. We evaluate our approach on several meteorological ensembles and include an assessment by domain experts.

## 2 RELATED WORK

Uncertainty visualization, one of the top challenges in scientific visualization [Bon+14, PRJ12], is often estimated by ensembles – representative samples of realizations of simulated phenomena, obtained by running simulations with different initial conditions and models. Such data is typically spatio-temporal, multivariate, and multivalued [KH13, LPK05], making its analysis and visualization difficult. Typical methods evaluate summary statistics and visualize these using color maps, contours, surface deformation, opacity, boxplots, or glyphs [Pot+09, LPK05, PMW13, PKRJ10].

For vector fields, Wittenbrink et al. [WPL96] propose glyphs to show the magnitude and angular uncertainty. Lodha et al. [LPSW96] show the flow uncertainty using envelopes and animation. Pfaffelmoser et al. [PMW13] present circular glyphs for the uncertainty of gradients in mean and orientation in 2D scalar fields. Jarema et al. [JDKW15] use lobular glyphs to visualize multimodal distributions for 2D directional data. Other local methods include texture mapping [BWE05] and a reaction-diffusion model [SJK04]. Allendes Osorio and Brodlie [AOB09] adapt LIC for 2D uncertain steady vectors. For time-varying uncertain vector fields, Hlawatsch et al. [HLNW11] introduce flow radar glyphs. In the crisp case, Hlawatsch et al. [HSJW14] downscale individual pathlines, while we downscale ensembles of trajectories. To consider the transport uncertainty, Otto et al. use particle density functions to obtain an uncertain topological segmentation of 2D [OGHT10] and 3D [OGT11] Gaussian-distributed steady vector fields; Schneider et al. [SFRS12] analyze the transport uncertainty for unsteady vector fields. Hummel et al. [HOGJ13] compare the material transport in time-varying flow ensembles by computing individual and joint vector field variances. We analyze the transport variability based on trajectory dissimilarities, rather than variances, which allows us to identify when divergences occur.

Clustering is another method for dealing with large and complex data [Jai10]. Bruckner and Möller [BM10] propose density-based clustering to identify similar volumetric time sequences in physically-based ensemble simulations. Bordoloi et al. [BKS04] perform realization- and distribution-based hierarchical clustering of ensemble data. Hierarchical clustering is also used to cluster trajectories, e.g., for blood flow [Oel+14] or meteorological data [FBW16]. Hollister and Pang [HP16] cluster streamlines using DBSCAN to derive scalar fields capturing cases when ensemble members may exhibit strong separation along trajectories, but a weak terminal separation. Although our analysis is performed on trajectories, it clusters trajectory positions and not the trajectories themselves. Thus, we are able to determine not only whether divergences occur along the trajectories, but also when they occur. Jarema et al. [JDKW15] use GMMs to cluster ensemble members based on the extent of their directional similarity locally. We extend this method to cluster ensemble members by their transport similarity, where the dissimilarity of two members is obtained by applying the Mahalanobis distance on the modes identified by means of a GMM approximation.

GMMs have been used before in uncertainty visualization. Correa et al. [CCM09] use GMMs to model uncertainty in the visual analysis process, while Hollister and Pang [HP13] apply GMMs to perform probability distribution interpolation. Liu et al. [LLBP12] employ multiple Gaussian components for a volume rendering of stochastic fields. The Mahalanobis distance [Mah36] has been used, e.g., for uncertain scalar fields, to assess the positional uncertainty of isosurfaces [PRW11] or reveal the possible locations of critical points [MW14].

Other techniques for ensemble data include coordinated multiple views, which study multivariate relations via linking and brushing [KH13], and parallel coordinates [HW13] or parallel sets [BKH05]. Nocke et al. [NFB07], and Molchanov and Linsen [ML14] use coordinated multiple views for climate ensem-

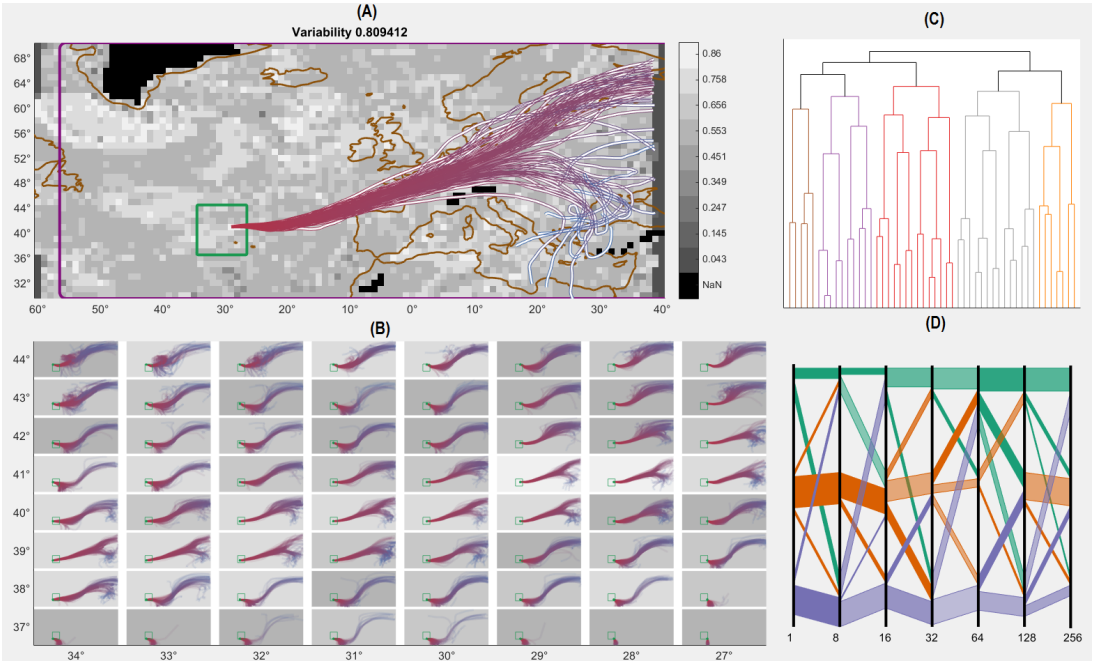



Figure 1: Multiple linked views for an ECMWF (European Centre for Medium-Range Weather Forecasts) time-varying ensemble. The spatial view (A) shows the aggregated flow variability over the domain and an enhanced spaghetti plot at a user-selected location. For the selected region marked by a green square, the detail view (B) shows downscaled spaghetti plots. View (C) shows the hierarchical clustering of the members with perturbed initial conditions. View (D) shows the variability of the clustering solution at selected time steps.

bles. Piringer et al. [PPBT12] analyze 2D function ensembles on three levels of details (member- and domain-oriented, and surface plot).

## 3 OVERVIEW

Our method starts with a vector field ensemble given on a 2D grid structure, and identifies at every grid point the time steps and locations of dissimilar behavior. From this, we derive measures for the temporal evolution of the transport variability, to show the variability over the domain (cf. background in Fig. 1(A)) and generate enhanced spaghetti plots at selected locations (cf. foreground in Fig. 1(A)). The enhanced plots reveal the flow variability even when particles follow geometrically similar trajectories, but with significantly different speeds, being temporally actually dispersed.

To explore and compare the flow variability at multiple locations simultaneously, concurrent spaghetti plots lead to massive clutter and occlusion. Hlawatsch et al. [HSJW14] juxtapose miniaturized trajectory images to overcome this limitation for crisp vector fields. Our ensemble visualization builds upon the concept of small-multiples [Tuf83], but includes the derived transport variability to construct small-multiples preserving the main trends of the particle trajectories (cf. Fig. 1(B)). This is achieved by downscaling trajectories based on a selection of salient time steps.

Finally, our method clusters members based on their global transport similarity, and shows these clusters via dendrograms (cf. Fig. 1(C)). The temporal evolution of clusters is encoded visually via parallel sets (cf. Fig. 1(D)), from which splitting and merging events over time can be deduced.

To quantitatively assess the flow variability, our method determines when and where members behave dissimilarly, without imposing any synthetic thresholds beyond which the behavior is considered dissimilar (cf. Sec. 4.1). This allows us to assess the spatial variability at individual time steps or over a forecast interval, and obtain the salient time steps for the small-multiples layout (cf. Sec. 4.2). We also use the method to cluster ensemble members based on their transport similarity throughout the domain (cf. Sec. 4.3).

## 4 TRANSPORT VARIABILITY

To find out when and where changes in the transport behavior occur, we assess members as (dis)similar based on their deviation at every time step, but without imposing any artificial thresholds. For normally distributed data, the Mahalanobis distance [Mah36] specifies natural thresholds. Due to its use of the covariance matrix, it is scale-independent and unaffected by correlations between variables. Members are thus (dis)similar based on their deviation relative to the data variability. Ensemble data is, however, often not Gaussian distributed. We thus model the distribution of 2D particle positions (seeded at the same point) at every time step with a mixture of Gaussians, and apply the Mahalanobis distance

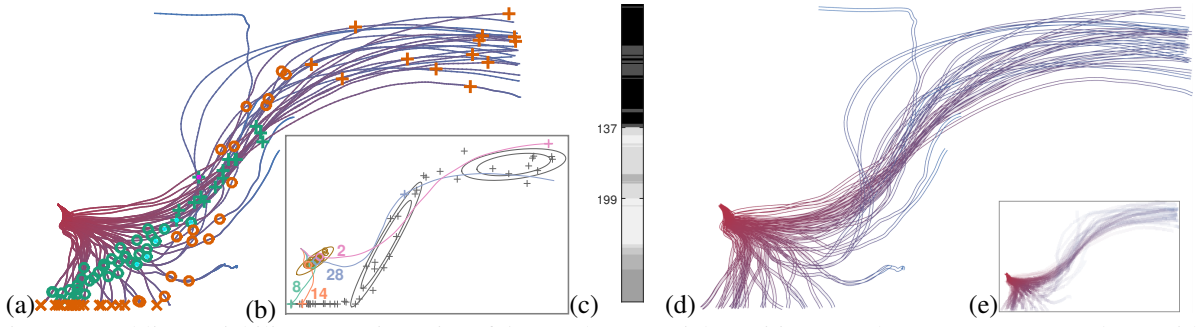



Figure 2: Pathline variability: (a) trajectories of 51 members; particle positions are shown at steps 137 and 199 with green and orange markers, respectively, and different symbols for different clusters; (b) four singled-out members: 8 (green), 14 (orange), 28 (blue), and 2 (pink), with markers for all members and ellipses for the Gaussian modes at steps 58 and 199; (c) grayscale encoding of the time step divergence count; as the divergence counts increase, the stripes (d) fade to white in the enhanced spaghetti plot and (e) become less opaque in the downscaled version.

to assess pairs of members as (dis)similar relative to the variability of the corresponding mode.

The intuition behind this fine-grained analysis is illustrated in Fig. 2(b) for four selected members of an ensemble: members 2, 8, 14, and 28. Even though a bimodal distribution is fitted only at step 137 and no synthetic threshold is defined for the spread, member 8, initially an inlier, is, up to step 58, similar only to member 28 relative to the local variability (step 58 is marked by brown concentric ellipses for the Gaussian mode and circles for the particle positions). Thereafter, it also becomes and stays similar to member 14, both following the downward trend. As the deviation between members 8 and 28 increases, the two become dissimilar shortly after step 58; members 8 and 2 are dissimilar from the very beginning, despite the spread being initially very low. Actually, member 2 is dissimilar for most of the time even to 28, although they both show geometrically similar trajectories and follow the upward trend. This happens because the particles travel at different speeds (at, e.g., step 199, where the three fitted modes are shown by concentric gray ellipses and all members are marked with pluses, note that positions are modeled by different modes).

## 4.1 Pairwise Dissimilarity Analysis

We consider flow field ensembles – collections of $n$ vector fields defined over the same grid structure. Our approach is designed for both stationary and time-dependent 2D vector fields, but the analysis can be extended directly to 3D. We obtain the trajectories by numerical integration, e.g., using 4-th-order Runge-Kutta methods. At every grid point, each of the $n$ trajectories comprises $m_i$ integration steps. To make all trajectories of equal length $m$, we repeat the final positions $m-m_i$ times, where $m=\max_{i=1,n}(m_i)$. Otherwise, members with similar trajectories, but of slightly different lengths $m_i > m_j$, e.g., members 8 and 14 in Fig. 2(b), would be artificially dissimilar in the $m_i - m_j$ interval. Members with trajectories of considerably different length and behavior are dissimilar as soon as their deviations are large relative to the allowed variability or the positions are modeled by different modes, e.g., members 8 and 28 in the same figure. Moreover, the (dis)similarity of two members $(i,j)$ is fixed after step $\max(m_i,m_j)$.

The pairwise analysis occurs in two stages, performed at every grid point and integration step of the member trajectories seeded at that grid point: In the first stage, we use GMMs to model the distribution of particle positions at each time step. We determine both the number of modes and their shapes automatically, by adapting for 2D data the procedure described in [JDKW15] for 2D directional data. Thus, unless the positions can be assumed Gaussian distributed, an Expectation Maximization (EM) algorithm fits two Gaussian modes to the positions and assigns each member to the mode that is more likely to model the observation. The process repeats until the members in each partition can be assumed Gaussian distributed (cf. Fig. 2(b) for an example of GMM partitions – shown using concentric ellipses – at several time steps). Depending on the initial conditions, EM algorithms may lead to non-repetitive solutions. To alleviate this problem, we run the GMM algorithm several times with different starting values, and use silhouettes – validation techniques for algorithms that use random initial guesses – to select the best solution from those having the most frequently met modality.

A sample of 2D observations $x_i=[X_i,Y_i]$, $i=1,n$, is modeled with a mixture of $N$ Gaussian modes

$$f(x)=\sum_{j=1}^{N}\alpha_j\mathcal{N}(\mu_j,\Sigma_j),\ \ \alpha_j>0,\ \ \sum_{j=1}^{N}\alpha_j=1, \quad (1)$$

where each Gaussian mode $\mathcal{N}(\mu_j,\Sigma_j)$ has a weight $\alpha_j$ and is parameterized by its mean $\mu_j$ and covariance matrix $\Sigma_j$. Each observation $x_i, i=1,n$, is modeled by Gaussian mode $j$ with probability $p_{j,i}$, where $j=1,N$ and $\sum_{j=1}^{N}p_{j,i}=1$. The GMM creates a first partitioning of the members, depending on which mode is more

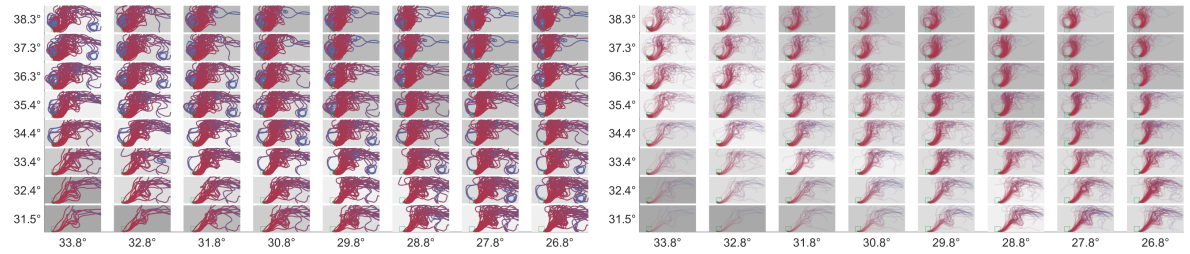



Figure 3: Small-multiples displays for ensembles are significantly more cluttered when downscaled in (left) a straightforward manner than (right) with sampling and variability encoding in opacity.

likely to model the observations. We use *soft* clustering, where observations are dissimilar only if they belong with high probabilities to different modes. Thus, observations located on the boundary of two modes are similar to observations modeled by both modes, as shown in Fig. 2(a): a bimodal distribution is fitted for the first time at step 137 (cf. the green markers, of two types, circles and pluses); a few members (with cyan dots in the middle of the markers) are at the boundaries of both modes and thus similar to all members. However, just because only at this step there is enough proof for bimodality, this does not necessarily mean that members now modeled by different modes have started diverging at this step. Their deviation may have been large relative to the local variability already earlier, e.g., members 8 and 28, or 8 and 2 in Fig. 2(b).

Thus, in the second stage, for every pair of similar members $(x_i, x_j)$, we compute the Mahalanobis distance relative to the corresponding covariance matrix $\Sigma_k$

$$MD(x_i,x_j) = \sqrt{(x_i - x_j)\Sigma_k^{-1}(x_i - x_j)^{\mathrm{T}}}. \qquad (2)$$

Members are dissimilar if $MD(x_i,x_j) > 2.3$, which corresponds statistically to less likely deviations falling outside one confidence region (68%) for a bivariate Gaussian distribution $\mathcal{N}(\mu_k,\Sigma_k)$. For boundary members, we compute the distance relative to both modes and take the higher value. Out of the six members in the previous example, only the one in the upper mode still exhibits similarities to members in the other mode. This occurs because the covariance matrix allows more variation in the vertical direction than in the horizontal one. For the same reason, the member with a magenta dot in the middle is dissimilar to all others.

## 4.2 Variability over the Domain

The analysis based on GMMs and the Mahalanobis distance identifies the pairwise (dis)similarities of the ensemble members at every time step. To distinguish between inliers and outliers within modes or among all members, we define the *member divergence count* ($\mathrm{mdc}_i^t$) of ensemble member $i$ at step $t$ as the number of dissimilar members at that instant, normalized by the total number of members. For Gaussian distributions, positions closer to the mean have lower divergence counts than those located further. For multimodal distributions, the divergence counts generally increase for both mode inliers and outliers.

To reveal the time steps and locations with higher flow variability, we sum up and normalize the divergence counts of all members at a time step $t_k$ to obtain a *time step divergence count* ($\mathrm{tdc}^{t_k}$). The aggregated time step divergence count over a given time interval $[t_1,t_2]$ is $\mathrm{median}(\mathrm{tdc}^{t_k})_{t_k\in[t_1,t_2]}$. We use the median because it is a robust measure of the central tendency of the data, but other summary statistics could be used instead. Moreover, to distinguish between low and high divergence counts, we define reference values, per member (the median of the minimum divergence count at every grid point $(i,j)$, $\mathrm{median}(\min(\mathrm{mdc}_{i,j_k^t})_{k=1,n,t=1,t_{\max}})$), and per time step (the median of the divergence counts at the first time step, $\mathrm{median}(\mathrm{tdc}^1_{i,j})$).

We also use the divergence counts to find a selection of time steps that preserves the transport behavior. Thus, we downscale trajectories without obscuring the transport trends and variability, as a straightforward downscaling would (cf. Fig. 3 (left)). For all members, we keep the salient trajectory points – the steps with changes in (dis)similarities. Other steps are sampled regularly, the sampling rate being proportional to the number of non-salient steps. While the information loss in intervals with high curvature may lead to a coarser curve approximation, the flow structure is qualitatively well-preserved (cf. Fig. 3 (right)).

The time step divergence counts for the previous example (with values in the $[0.55, 0.82]$ range) are encoded in grayscale in Fig. 2(c). While there is not enough proof for a departure from normality, the divergence counts vary in the $[0.55, 0.61]$ range. A surge to 0.76 occurs at step 137, when a bimodal distribution is fitted to the data, succeeded by a small gradual decreases as more particles follow the upward trend. Despite the geometrical similarity of the trajectories going upwards, particles move at different speeds. This leads to a second surge (0.81) at step 199, when three Gaussian modes are fitted (cf. the orange markers in Fig. 2(a)). As particles approach their final positions (and since most of them end up together, either in the bottom or the right bundle), the divergence count decreases again, although to values no lower than 0.69. During downscaling, the non-salient time steps are sampled and around 60% of

the trajectory points discarded, but the flow structure and variability are retained (cf. Fig. 2(e)).

### 4.3 Flow-based Similarity

To analyze the transport similarity of the ensemble, we extend the work of Jarema et al. [JDKW15], who model directional data locally via GMMs and use the modes to cluster members hierarchically based on their local angular similarity over the whole domain. The binary similarity of any two members at a point depends on the angular deviation relative to local variation of their mode, and the global similarity measure is defined as the percentage of similar locations out of the total number of grid points. Here, we use the Mahalanobis distance to determine deviations in particle positions (rather than angles) that are statistically meaningful relative to the mode variability. Moreover, we perform the clustering over a whole forecast interval to cluster ensemble members based on their transport similarity (rather than their local angular similarity).

## 5 VISUAL ANALYSIS OF THE TRANSPORT VARIABILITY

Our user interface (cf. Fig. 1) comprises four linked views, two for the variability over the domain and another two for the variability of the clustering solution.

### 5.1 Flow Variability over the Domain

The spatial view (cf. Fig. 1(A)) shows the ensemble variability (here the aggregated time step divergence count for the entire forecast interval and all members) over the domain using a sequential grayscale colormap. A time slider allows an interactive visualization of the variability for single time steps or aggregated over time intervals, either for a selection of members or the entire ensemble (by using the time step divergence count). Users can select locations where to display enhanced spaghetti plots. Grayscale colormaps permit the use of a colorblind-safe red-to-blue variation for the trajectories, making them stand out against the background. Nevertheless, they suffer from simultaneous contrast effects, which hinders the correct interpretation of the displayed data [War13]. To ease the variability evaluation, we inform the user about the variability at the selected location and show the global reference values.

Trajectory plots encode both the integration time and the variability (member divergence counts). Each trajectory is displayed as a curve stripe, the contour of the stripe color-coding the time and the stripe itself the variability. The color of each stripe is based on that of its contour, fading to white as the member divergence count increases relative to the local reference value (the smaller value between the local minimum and the global reference value). Thus, the stripe color for members having divergence counts close to the local reference value can hardly be distinguished from that of the contour. As the divergence count increases, the stripe color fades to white, making the trajectories of members with higher variability stand out less against the grayscale background than those with lower variability. The time information is nevertheless fully preserved in the color of the contours.

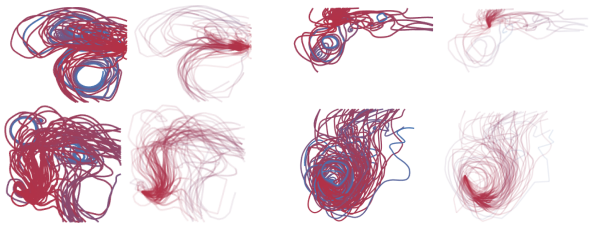


Figure 4: In a straightforward downscaling (odd plots – the first plot is the leftmost top plot) clutter and occlusion obscure the flow trends and variability; downscaling with sampling and variability encoding (even plots) conveys the main flow patterns and variability.

Fig. 2(d) shows the enhanced spaghetti plot for the previous example. The stripes begin to fade rapidly, as particles tend to spread out from an early stage. However, up to step 137, when a bimodal distribution is fitted to the data, the color of most stripes, albeit suggestive of the proximate split, has yet to fade completely. Thereafter, the stripes of several trajectories going upwards fade to white, because particles travel at considerably different velocities and member divergence counts are high. Despite the geometrical similarity of the trajectories, the fading reveals the flow variability (the increasing dispersion of the particles and number of partitions). Also, the stripes fade less for most trajectories in the lower bundle; their flow behavior is more similar, even if from their geometry alone they appear more dispersed than the trajectories in the upper bundle.

### 5.2 Flow Variability over Subdomains

To allow the concurrent visualization of enhanced spaghetti plots across selected subdomains with significantly less clutter and occlusion, we propose a small-multiples layout with miniaturized plots, where the downscaling preserves the salient flow trends.

The detail view (cf. Fig. 1(B)) shows a predefined number of downscaled plots, computed at regular intervals in the domain. A straightforward downscaling may obstruct the flow trends and variability (cf. Fig. 4). For example, in the first plot, the ending spiraling structures of several pathlines obscure the stable flow pattern at the beginning. Instead, we determine a selection of time steps that preserves the transport behavior (cf. Section 4.2). We also encode the integration time directly in the color of the stripes and the variability in the opacity, so that the main flow trends are clearly discernible in the miniaturized versions. Notice how the opacity use

and the dense sampling of the spiraling parts (where the flow variability hardly changes anymore) in the second plot in Fig. 4 help bring out both the flow patterns and the variability.

To enable a comparative visual analysis of the downscaled plots, all plots have the aspect ratio of the original domain and the same domain section (shown in the main spatial view by a purple rectangle). The area occupied by the detail view is displayed by a green square. To preserve context information, the green square (correspondingly scaled) is also shown in each downscaled plot, along with a green anchor ball that marks the seeding point of the trajectories. The initial domain section is the bounding box containing all spaghetti plots, but can be adjusted interactively (as was done in Fig. 1(A)).

### 5.3 Flow-based Similarity Visualization

The hierarchical clustering of the ensemble (based on the global flow similarity over the whole time interval) is summarized as a dendogram (cf. Fig. 1(C)). The ensemble members are shown on the horizontal axis and the clustering levels on the vertical axis. The dissimilarity levels increase as subclusters are merged, joins being represented graphically as inverted U lines. Neighboring larger subclusters are shown in different colors, to make partitions stand out. Users can select (groups) of members in the dendogram (or as text input) to visualize their spatial variability.

Insight into the dynamical evolution of the clustering solution is gained using parallel sets [BKH05] (cf. Fig. 1(D), with a selection of time steps along the horizontal axis and the cluster ids along the vertical one). We perform a hierarchical clustering at every selected time step. For the sake of uniformity and simplicity, we partition the members into at most three clusters, and track how the clustering solution varies over the selected time steps. We sample the time steps more densely at the beginning, e.g., in powers of two, because we noticed that the divergence counts and local clustering solutions vary more earlier in the integration (due, e.g., to changes in flow regime occurring early). The clustering variability is shown using branches connecting clusters at consecutive selected steps, the thickness of each branch depending on the corresponding number of members. Clusters are ordered so as to support id continuity from one step to another. Thus, we compute the number of common members for every pair of clusters at consecutive time steps, and assign cluster ids in decreasing order. To facilitate tracking the split and join events, bars starting from the same cluster have the same color: green, orange, and purple for the first, second, and third cluster, respectively.

This encoding reveals how the clustering solution varies from one time step to another, but not how the cluster memberships vary. To gain insight into this kind of variability, we determine, for the members in each branch, the cluster ids at the previous step and compute their cardinalities. We then map the ratio of the largest cardinality to the number of members, to the opacity of each branch. The more opaque the branch is, the less the cluster componence has changed. To shed further light onto the clustering variability, users can select a branch to view the clustering evolution over all selected steps for the members in the selected branch (cf. Fig. 6).

## 6 RESULTS

We illustrate our framework on two ECMWF (European Centre for Medium-Range Weather Forecasts) ensembles of dimensions $101 \times 41$, each comprising a control run and another 50 members with perturbed initial conditions. Details on the system are available in [LP08].

### 6.1 Transport Variability Analysis

The first ensemble is a time-varying wind forecast at a pressure level of 850 hPa, initiated on October 15, 2012. The geolocated variability field (aggregated over the entire forecast interval for all members) is shown in Fig. 1(A). Locations where data was not available at this pressure level have been marked as "Not a Number (NaN)". We noticed that pathlines seeded over heterogeneous color regions (cf. Fig. 1(B)) exhibit dissimilar flow behavior, as opposed to those seeded over homogeneous areas. The detail view enables a comparative visual analysis of the pathlines, revealing the numerous changes in flow patterns and variability across the subdomain. In Fig. 1(B), for instance, trajectories seeded in the bottom line (to the left), where most pathlines go southwards, display aggregated divergence counts similar to the time step reference value (0.55). The variability increases to the right (to 0.65) as more particles go northeastwards, and decreases again further to the right, where most pathlines show no change of regime. In the next row (to the left), most particles follow a northeastward trend, but with various velocities and increasing dispersion, in spite of the geometrical similarity of the trajectories (values are around 0.7). The variability increases further (close to 0.8) in the third row, where the trajectories bifurcate geometrically as well.

The clustering solution of the ensemble is shown in Fig. 1(C) for the members with perturbed initial conditions. At first, we performed a hierarchical clustering of all members. However, because the control run was very similar to the majority of the other members, the large cluster forming around the dendogram hindered the identification of natural divisions in the hierarchical tree. Upon excluding the control run, we could detect natural groupings in the dendogram, which revealed an

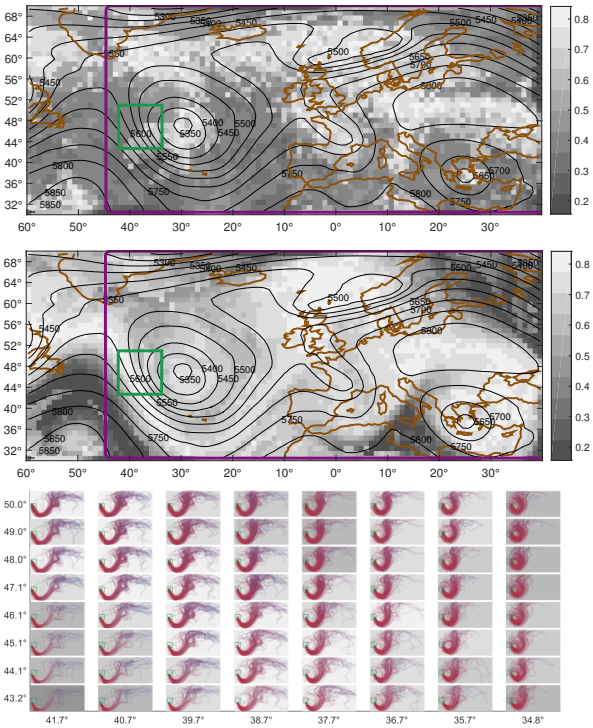






Figure 5: Spatial view for stationary ECMWF ensemble showing the variability at the (top) first time step and (middle) over the whole time interval. (Bottom) Detail view with downscaled streamline plots.

interesting pattern that should be further explored together with domain experts: ensemble member $i$ was typically considerably more similar to ensemble members $i+20$ and, if available, $i+40$, than to other members. Notice that, in the dendogram, groupings of two or three members predominate at the low levels, larger clusters forming at distinctly higher levels.

The second ensemble is a stationary wind forecast ensemble at the same pressure level, valid on October 19, 2012. The geolocated variability for all members is displayed over the entire domain in Fig. 5, for the first time step (top) and aggregated over the whole forecast interval (middle). The isocontours of the geopotential height field of the control run are shown as black contours. Initially, regions of high variability are found mostly at the pressure centers, but their extent increases over time, as particles seeded in regions of lower variability enter regions of higher variability. This is also noticeable in the detail view (bottom), where, at all locations, most streamlines follow the southeastwards-northeastwards trend of the isocontours before they begin to spread out. In fact, the variability in this region remains relatively low for much longer than in other regions. In the end, the regions with low variability occupy much smaller extents, e.g., the low-frequency elongated region of high pressure in the bottom left corner.

The clustering solution for this ensemble considers all members. Fig. 6 (top) shows its time variability at seven

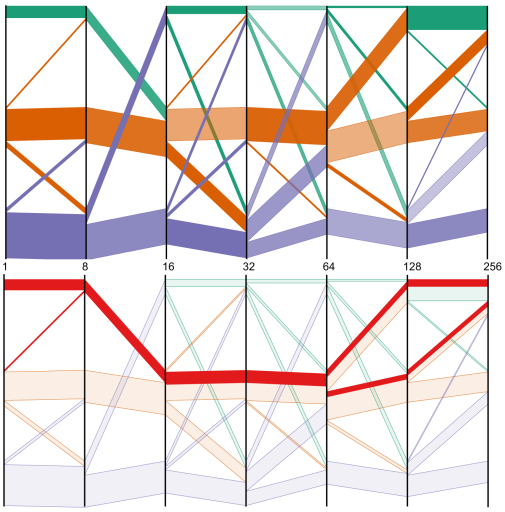


Figure 6: (Top) Clustering variability at selected time steps (cluster ids on the vertical axis, time steps on the horizontal axis). (Bottom) Membership variability for selected subcluster.

selected time steps. The clustering solutions appear quite stable at the first two steps, but the variability increases afterwards. The opacity of the branches gives a first insight into the cluster membership variability. For instance, the thickest of the orange branches, joining the second clusters at selected steps 16 and 32, is quite transparent, implying that the members in the branch are a mixture of those in the previous green and orange branches. The orange branch joining the second and third clusters is, however, opaque, as the componence of its members has not changed from the previous step. To gain further insight into the membership variability, users can select branches to see how the cluster memberships of their members vary at the selected steps. Fig. 6 (bottom) shows an example for the green branch joining the second cluster at step 16. Observe that, except for a temporary split towards the end, the members in the branch are always clustered together.

## 6.2 Implementation, Performance Analysis, and Scalability

We ran our tests on a standard desktop PC, equipped with an Intel i7-4790 quad-core processor running at 3.6 GHz and with 12 GB RAM. Fitting GMMs and computing the pairwise (dis)similarities using the Mahalanobis distance can be performed in parallel at every time step and every grid point. The operations can thus be parallelized in a straightforward way on the GPU, resulting in computation times of under five seconds for all datasets. As shown in the accompanying video, we are able to handle the comparative analysis session at interactive frame rates.

Our approach is scalable for ensembles larger than those used in our examples. Although the readability of the dendogram may suffer if the number of members

is larger than a certain limit, 51 ensemble members, like in our case, is a typical number for meteorological ensembles. Similarly, the readability of the parallel coordinates depends on the number of selected time steps and may require focus and context techniques, such as zooming and panning, which are part of our future work. An increase in the grid dimensions affects neither the two views for the clustering variability, nor the detail view, since the number of downscaled plots displayed is fixed, but only the spatial view, where the size of the pixel of each grid point decreases.

# 7 CONCLUSION AND EVALUATION

In this paper, we proposed a novel framework that enables an interactive comparative visual analysis of the transport variability of ensembles of 2D vector fields. We showed that our approach is able to determine the pairwise (dis)similarities of the ensemble members at every time step, without imposing artificial thresholds for the deviations. To achieve this, we computed the pairwise dissimilarities using the Mahalanobis distance on the Gaussian components identified by a GMM algorithm. Based on this fine-grained analysis, we proposed means to convey the variability of the spatio-temporal evolution of an ensemble and its clustering solution.

In developing our techniques, we collaborated closely with domain experts from meteorology, and provide herewith a summary of their informal feedback. The experts found the proposed methods useful in gaining a fast insight into the flow predictability over the domain and the time interval over which the forecasts can be trusted. They also described the techniques as useful in determining regions of different qualitative flow and highly appreciated our small-multiples approach that displayed the flow behavior at several locations concurrently. Nevertheless, they were initially puzzled about certain neighboring locations where the flow looked similar, but the variability values were quite different. Detailed inspections of such cases revealed the dissimilarities in flow behavior that had lead to the different variability values, although occasionally the differences were caused by a suboptimal partitioning of the GMM algorithm. The experts were also surprised by the spread of particle positions along geometrically similar trajectories and commended the enhanced spaghetti plots for bringing out the variability in the flow. Regarding the clustering solution, they were interested to further investigate the reason for the pattern present in the clustering solution for the time-varying data.

In the future, we plan to enrich the possibilities of exploring the variability of the clustering solution and the dynamics of the ensemble. We also intend to extend our analysis to 3D vector data. While the mathematical extension is straightforward, novel graphical abstractions are necessary to reveal the ensemble variability that reduce the clutter and occlusion problems inherent to 3D.

# 8 ACKNOWLEDGMENTS

Access to ECMWF prediction data has been kindly provided in the context of the ECMWF special project "Support Tool for HALO Missions." We are grateful to the special project members Marc Rautenhaus and Andreas Dörnbrack for providing the ECMWF ENS datasets used in this study. The work was partly funded by the European Union under the ERC Advanced Grant 291372: Safer-Vis – Uncertainty Visualization for Reliable Data Discovery.

# 9 REFERENCES

[AOB09] Allendes Osorio, R. and Brodlie, K. W. Uncertain flow visualization using LIC. *Theory and Practice of Computer Graphics, Eurographics UK Chapter Proceedings*, pages 215–222, 2009.

[BKH05] Bendix, F., Kosara, R., and Hauser, H. Parallel sets: visual analysis of categorical data. In *IEEE Symposium on Information Visualization*, pages 133–140, 2005.

[Bon+14] Bonneau, G.-P., Hege, H.-C., Johnson, C. R., Oliveira, M. M., Potter, K., Rheingans, P., and Schultz, T. Overview and state-of-the-art of uncertainty visualization. In *Scientific Visualization*, pages 3–27. Springer, 2014.

[BKS04] Bordoloi, U., Kao, D., and Shen, H.-W. Visualization techniques for spatial probability density function data. *Data Sci. J.*, 3:153–162, 2004.

[BWE05] Botchen, R. P., Weiskopf, D., and Ertl, T. Texture-based visualization of uncertainty in flow fields. In *Proc. IEEE Visualization*, pages 647–654. IEEE, 2005.

[BM10] Bruckner, S. and Möller, T. Result-driven exploration of simulation parameter spaces for visual effects design. *IEEE TVCG*, 16(6):1468–1476, 2010.

[CCM09] Correa, C., Chan, Y.-H., and Ma, K.-L. A framework for uncertainty-aware visual analytics. In *Proc. IEEE VAST*, pages 51–58, 2009.

[FBW16] Ferstl, F., Bürger, K., and Westermann, R. Streamline variability plots for characterizing the uncertainty in vector field ensembles. *IEEE TVCG*, 22(1):767–776, 2016. ISSN 1077-2626.

[HW13] Heinrich, J. and Weiskopf, D. State of the art of parallel coordinates. *STAR Proceedings of Eurographics*, pages 95–116, 2013.

[HSJW14] Hlawatsch, M., Sadlo, F., Jang, H., and Weiskopf, D. Pathline glyphs. *Comput. Graph. Forum*, 33(2):497–506, May 2014. ISSN 0167-7055.

[HLNW11] Hlawatsch, M., Leube, P., Nowak, W., and Weiskopf, D. Flow radar glyphs & static visualization of unsteady flow with uncertainty. *IEEE TVCG*, 17(12):1949–1958, 2011.

[HP16] Hollister, B. E. and Pang, A. Visual Analysis of Transport Similarity in 2D CFD Ensembles. In *IS&T Electronic Imaging Conference on Visualization and Data Analysis*, pages 1–11, 2016.

[HP13] Hollister, B. E. and Pang, A. Interpolation of non-Gaussian probability distributions for ensemble visualization. Technical report, Jack Baskin School of Engineering, UC Santa Cruz, 2013.

[HOGJ13] Hummel, M., Obermaier, H., Garth, C., and Joy, K. Comparative visual analysis of Lagrangian transport in CFD ensembles. *IEEE TVCG*, 19(12): 2743–2752, 2013. ISSN 1077-2626.

[Jai10] Jain, A. K. Data clustering: 50 years beyond K-means. *Pattern Recognition Letters*, 31(8): 651–666, 2010.

[JDKW15] Jarema, M., Demir, I., Kehrer, J., and Westermann, R. Comparative visual analysis of vector field ensembles. In *Proc. IEEE VAST*, pages 81–88, 2015.

[KH13] Kehrer, J. and Hauser, H. Visualization and visual analysis of multi-faceted scientific data: A survey. *IEEE TVCG*, 19(3):495–513, 2013.

[Pot+09] Kristin, P. et al. Visualization of uncertainty and ensemble data: Exploration of climate modeling and weather forecast data with integrated ViSUS-CDAT systems. *J. Phys.: Conf. Ser.*, 180: 012089, 2009.

[LP08] Leutbecher, M. and Palmer, T. Ensemble forecasting. *Journal of Computational Physics*, 227 (7):3515–3539, 2008.

[LLBP12] Liu, S., Levine, J., Bremer, P., and Pascucci, V. Gaussian mixture model based volume visualization. In *Proc. LDAV*, pages 73–77, Oct 2012.

[LPSW96] Lodha, S. K., Pang, A., Sheehan, R. E., and Wittenbrink, C. M. UFLOW: Visualizing uncertainty in fluid flow. In *Proc. IEEE Visualization*, pages 249–254, 1996. ISBN 0-89791-864-9.

[LPK05] Love, A., Pang, A., and Kao, D. Visualizing spatial multivalue data. *IEEE Comput. Graph. Appl.*, 25(3):69–79, 2005.

[Mah36] Mahalanobis, P. C. On the generalised distance in statistics. *National Institute of Sciences of India*, 2(1):49–55, 1936.

[MW14] Mihai, M. and Westermann, R. Visualizing the stability of critical points in uncertain scalar fields. *Computers& Graphics*, 41:13 – 25, 2014. ISSN 0097-8493.

[ML14] Molchanov, V. and Linsen, L. Visual exploration of patterns in multi-run time-varying multifield simulation data using projected views. In *Proc. WSCG*, volume 21, pages 49–58, 2014.

[NFB07] Nocke, T., Flechsig, M., and Böhm, U. Visual exploration and evaluation of climate-related simulation data. In *Winter Simulation Conference*, pages 703–711, 2007.

[Oel+14] Oeltze, S. et al. Blood flow clustering and applications invirtual stenting of intracranial aneurysms. *IEEE TVCG*, 20(5):686–701, 2014.

[OGHT10] Otto, M., Germer, T., Hege, H.-C., and Theisel, H. Uncertain 2D vector field topology. *Comput. Graph. Forum*, 29(2):347–356, 2010. ISSN 1467-8659.

[OGT11] Otto, M., Germer, T., and Theisel, H. Uncertain topology of 3D vector fields. In *PacificVis*, pages 67–74, 2011.

[PRW11] Pfaffelmoser, T., Reitinger, M., and Westermann, R. Visualizing the positional and geometrical variability of isosurfaces in uncertain scalar fields. In *Comput. Graph. Forum*, volume 30, pages 951–960, 2011.

[PMW13] Pfaffelmoser, T., Mihai, M., and Westermann, R. Visualizing the variability of gradients in uncertain 2D scalar fields. *IEEE TVCG*, 19 (11):1948–1961, Nov 2013. ISSN 1077-2626.

[PPBT12] Piringer, H., Pajer, S., Berger, W., and Teichmann, H. Comparative visual analysis of 2D function ensembles. *Comput. Graph. Forum*, 31: 1195–1204, 2012. ISSN 1467-8659.

[PKRJ10] Potter, K., Kniss, J., Riesenfeld, R., and Johnson, C. Visualizing summary statistics and uncertainty. *Comput. Graph. Forum*, 29(3):823–832, 2010.

[PRJ12] Potter, K., Rosen, P., and Johnson, C. From quantification to visualization: A taxonomy of uncertainty visualization approaches. In *Uncertainty Quantification in Scientific Computing*, pages 226–249. 2012.

[SJK04] Sanderson, A. R., Johnson, C. R., and Kirby, R. M. Display of vector fields using a reaction-diffusion model. In *Proc. IEEE Visualization*, pages 115–122, 2004.

[SFRS12] Schneider, D., Fuhrmann, J., Reich, W., and Scheuermann, G. A variance based FTLE-like method for unsteady uncertain vector fields. In *Topological Methods in Data Analysis and Visualization II*, pages 255–268. Springer, 2012. ISBN 978-3-642-23174-2.

[Tuf83] Tufte, E. *The Visual Display of Quantitative Information*. Graphics Press, 1983.

[War13] Ware, C. *Information visualization: Perception for design*. Elsevier, 3rd edition, 2013.

[WPL96] Wittenbrink, C., Pang, A., and Lodha, S. Glyphs for visualizing uncertainty in vector fields. *IEEE TVCG*, 2(3):266–279, 1996.