FAKULTÄT
FÜR !NFORMATIK
Faculty of Informatics

Modeling Sources and Sinks in Crowded Scenes by Clustering Trajectory Points Obtained by Video-based Particle Advection

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Medieninformatik

eingereicht von

Mag. Rainer Planinc, BSc.
Matrikelnummer 0425163

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuung
Betreuerin: ao. Univ. Prof. Mag. Dipl.-Ing. Dr.techn. Margrit Gelautz
Mitwirkung: Dipl.-Ing. Dr.techn. Norbert Brändle (Austrian Institute of Technology)

Wien, 09.06.2010

(Unterschrift Verfasser/in)

(Unterschrift Betreuer/in)

Declaration of Authorship

I, Rainer Planinc, declare that this thesis titled, "Modeling Sources and Sinks in Crowded Scenes by Clustering Trajectory Points Obtained by Video-based Particle Advection" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: _____

Date: _____

"It is hard to fail, but it is worse never to have tried to succeed."

Theodore Roosevelt

Abstract

Analysis of real surveillance video footage is very challenging – hence this thesis provides solutions to enhance the quality of trajectories of dense crowded scenes in real-time. An efficient algorithm models dense crowded scenes with the aid of particles, moved by the optical flow calculated between two consecutive frames. Thus trajectories are obtained without using a people tracking algorithm. Sources and sinks are modeled by clustering of start and end points. As dense crowded scenes are analyzed, many trajectories are interrupted thus making the choice of an appropriate clustering algorithm challenging – this thesis provides approaches to enhance the quality of trajectories. Furthermore, it evaluates different clustering algorithms and their practicability in combination with the real-time particle advection algorithm on benchmark data of a Viennese train station and additional data provided by the PETS workshop and the University of Central Florida.

Kurzfassung

Die vorliegende Diplomarbeit stellt einen Ansatz zur Echtzeit-Analyse von Videoszenen mit dichten Menschenmassen vor. Dazu modelliert ein effizienter Algorithmus die Szene mit Hilfe von Partikeln, welche auf Grund des optischen Flusses zwischen zwei aufeinanderfolgenden Videoframes bewegt werden. Dadurch können Trajektorien ohne den Einsatz eines Tracking-Algorithmus gewonnen werden. Die Modellierung von interessanten Teilbereichen der Szene erfolgt in einem anschließenden Schritt.

Da es sich bei den Überwachungsvideos um Szenen mit dichten Menschenmengen handelt, sind viele Trajektorien unterbrochen und stellen so eine große Herausforderung für die richtige Wahl eines Clustering-Algorithmus dar – diese Arbeit stellt Lösungsansätze vor, um die Qualität der Trajektorien zu erhöhen. Zur Modellierung von Quellen und Senken werden einige bekannte Clustering Algorithmen und deren Anwendung für Überwachungsvideos in Kombination mit dem entwickelten Echtzeit Partikel Advektionsalgorithmus evaluiert, wobei für die Evaluierung Videos einer Wiener Bahnhofshalle und des PETS Workshops, als auch Videos der Universität von Florida verwendet wurden.

Acknowledgements

It is a pleasure to thank ao. Univ. Prof. Mag. DI Dr. Margrit Gelautz who made this thesis possible. I am very grateful to Dipl.-Ing. Dr.techn. Norbert Brändle, whose guidance and support enabled me to get a deep insight in really interesting topics and helped me to improve my skills.

I owe my deepest gratitude to my parents Gertrude and Otto, who supported me with the best as they could every day in my life and who made my academic studies possible. Finally, I would like to heartily thank Rebecca, who always had an open ear for me and who helped me to find the right work-life-balance.

Contents

List of Figures

Abbreviations

EM	**E**xpectation **M**aximization
GNG	**G**rowing **N**eural **G**as
GPU	**G**raphics **P**rocessing **U**nit
KDE	**K**ernel **D**ensity **E**stimator
MDL	**M**inimum **D**escription **L**ength
PETS	**P**erformance **E**valuation of **T**racking and **S**urveillance
UCF	**U**niversity of **C**entral **F**lorida

Dedicated to my family

Chapter 1

Introduction

Automated video analysis is used in many different environments to fulfill different objectives. It can be used to retain a high quality standard during a manufacturing process, as it detects faulty parts automatically. It also deals with object detection in order to retrieve higher level information from a video sequence. During the last years, automated video analysis developed further quickly. It facilitates crowd analysis and detects trends in human behaviors. This knowledge can be used to control and guide people flows, avoid safety-critical overcrowding and results in advices to improve orientation systems.

The area of research on automated video analysis in visual surveillance is huge, but most approaches are interested in detecting motion of people or vehicles. Therefore many approaches to detect the motion itself or people and vehicles have been developed. If people or vehicles are detected by a mechanism, the position or change of position of a person or vehicle over time is analyzed as long as the person or vehicle does not leave the camera field of view.

Definition 1.1. A **trajectory** is the spatio-temporal position change of an object due to motion.

One of the most interesting elements of a scene are areas where subjects come from or are heading to – such as doors, staircases or areas where subjects get into or leave the camera field of view. Figure 1.1 shows an example containing five areas of interest: two doors and three areas of entry to or exit from the camera field of view. The information about areas of interest already exists indirectly in trajectory data – the first and last points of a person's trajectory are the sources and the sinks for a specific subject. One main objective of this work is to cluster trajectory start and end points representing sources and sinks to obtain higher level information from a scene.

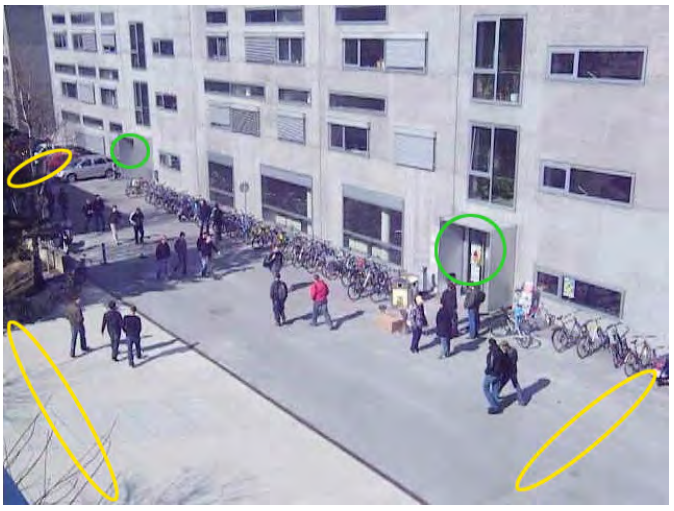


Figure 1.1: Areas of interest like doors (green) and entry to or exit from the camera field of view (yellow)

1.1 Challenges

A common approach to generate people trajectories is to track individual people. Tracking of individuals is very complex when using it with real surveillance videos, resulting in the following challenges:

- **Occlusions:** It is very important that the tracking algorithm be able to track the same person throughout the video. This can be very challenging in case of fully or partially occlusions as the tracking algorithm might follow another person after the occlusion occurred. Hence, some kind of additional knowledge to separate people after occlusions is required. An approach dealing with short time occlusions presented by [6] uses templates for subject tracking. The main feature to enhance stability is a mechanism which detects occlusions and their duration. If an occlusion is detected, the object is not tracked any more. To detect the end of the occlusion, the previously used template is matched with all frames from the beginning of occlusion to a maximum number of frames. The end of the occlusion is detected to be that frame where the best match with the template was found. After the end of the occlusion, the object is tracked again. Another approach presented in [7] does not only use templates but also visual features like color and texture. Occlusions are detected due to the rapid change in size of the occluded object. Recovering the occluded object contours facilitates tracking even during occlusions. A similar approach introduced by [8] is a segmentation algorithm for not tracking the entire object but parts of the object during severe occlusions.

- **Dense Crowded Scenes:** Most surveillance cameras are placed in public places like train stations, airports or streets. Hence, most of these public places are very

crowded thus making automated video analysis challenging and resulting in very complex approaches like in [9] – but most approaches often only work for loose groups of people. The work of [10] uses a Correlated Topic Model presented by [11] to generate high-level information in unstructured scenes by analyzing low-level optical flow vectors.

Definition 1.2. Optical Flow is referred to a vector field, where each vector represents the direction and amount of motion between two consecutive video frames.

The generated model is then incorporated into a tracking framework thus reducing the average tracking error. Another approach by [12] combines feature tracking and the optical flow to achieve long-range motion estimation. An optimization process repositions the particles – therefore the video is not analyzed linearly but nonlinearly. Neither [10] nor [12] are able to overcome the challenges of dense crowded scenes in real-time. For those who are interested in further approaches, a survey of different state-of-the-art tracking algorithms can be found in [13].

Definition 1.3. A **particle** is an image point moved by the underlying optical flow vectors, thus generating a trajectory.

Definition 1.4. The movement of a particle due to an optical flow vector is called **particle advection**.

Definition 1.5. If a particle only moves within a small radius over a specified time and number of frames, it is called **stranded** as it does not move any more.

This work provides an alternative to individual people tracking and is based on the efficient implementation of an optical flow field calculation by [14] facilitating real-time video analysis. The proposed approach calculates the optical flow field and particle advection in one step, thus saving time, memory space and enabling real-time particle advection. First, particles are arranged as a grid. Particles move then according to their corresponding motion vector until they either leave the frame or get stranded. Particle trajectories are obtained by moving the particles not only once, but frame by frame over a longer time period. To ensure that the number of particles remains at an equally high level, new particles are inserted after a particular time. In order to obtain an entire trajectory, the newly inserted particles are moved back in time to determine where they would come from if they had been inserted before.

Definition 1.6. Forward advection is referred to the process of advecting particles forward in time.

Definition 1.7. Backward advection is referred to the process of advecting particles backward in time.

1.2 Modeling Sources and Sinks

Analyzing the distribution of the start and endpoints (of a set of trajectories) yields the positions where subjects are likely to enter or leave the scene [15, 16]. Eliminating all trajectory points except the first and the last, results in two point sets – one for the sources, the other one for sinks. An approach presented by [17] obtains sources and sinks by trajectory clustering and analyzing the start and end of clustered paths, requiring considerably more computational work than our approach and probably having difficulties to cluster trajectories in unstructured dense crowded scenes.

Definition 1.8. Particle hopping is the process of a particle "jumping" from one person to another due to occlusion, thus distorting trajectories.

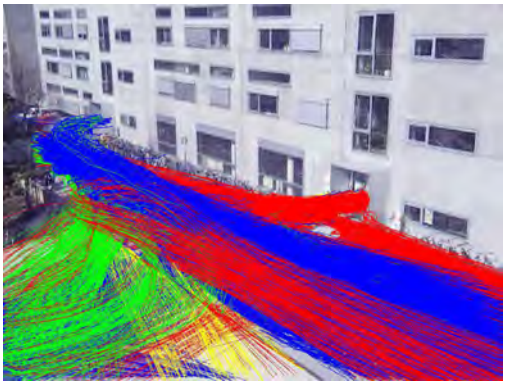
Figure 1.2a shows the particles during particle advection, the corresponding trajectories are depicted in Figure 1.2b. The correspondent density of start and end points is shown in Figure 1.2c and 1.2c. The approximate direction of people trajectories is coded by the use of different colors in Figure 1.2b. Red trajectories indicate people going from left to right, whereas blue trajectories indicate that people moved from the right to the left side. People moving from the top downwards are visualized by green trajectories, trajectories of people moving upwards are yellow. Colors used in Figure 1.2c and Figure 1.2d represent the density of start and end points - cold colors (e.g. blue) visualize small densities. The warmer the color gets (e.g. green, yellow, red) the more start and end points are placed in this area.

Distinction between "real" sources or sinks and sources or sinks caused by noise, particle hopping or incomplete trajectories is very challenging. This task is approached by dividing the start and end points into different clusters. Assuming that most trajectories start at a source cluster and end at a sink cluster, dense clusters are more likely to be a source or sink than clusters with widely distributed data points. Thus, we consider compact clusters as a good evidence of correctly identifying entry and exit zones. The information about sources and sinks in a surveillance video facilitates a fundamental knowledge about the scene structure, resulting in enhanced trajectories and the possibility of detecting unusual activities.

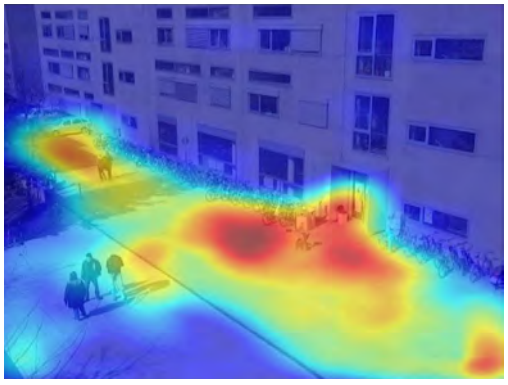
1.3 Objectives

The objectives of this master thesis can be described as follows:

(a) Particles during particle advection



(b) Trajectories obtained by particle advection



(c) Density of start points

(d) Density of end points

Figure 1.2: Sources and sinks obtained by trajectories

- **Objective 1:** to develop a technique for particle advection introduced in Definition 1.4 and to overcome the challenges of stranding particles beyond sources and sinks (due to obstacles) and particle hopping (due to occlusions).
- **Objective 2:** to implement the developed algorithm in an efficient manner, hence facilitating real-time video analysis.
- **Objective 3:** to model sources and sinks using an appropriate clustering algorithm. Ideally, all trajectories should start in sources and end in sinks, but in practice a criterion for distinguishing between true and wrong sources and sinks has to be found.
- **Objective 4:** to evaluate the implemented particle advection and clustering algorithm on crowded benchmark videos provided by PETS (Performance Evaluation of Tracking and Surveillance) workshop, the University of Central Florida (UCF), and dense crowded videos of a Viennese train station.

The flowchart depicted in Figure 1.3 shows the workflow of the proposed framework including inputs and outputs of each step. This thesis is organized as follows: Chapter 2

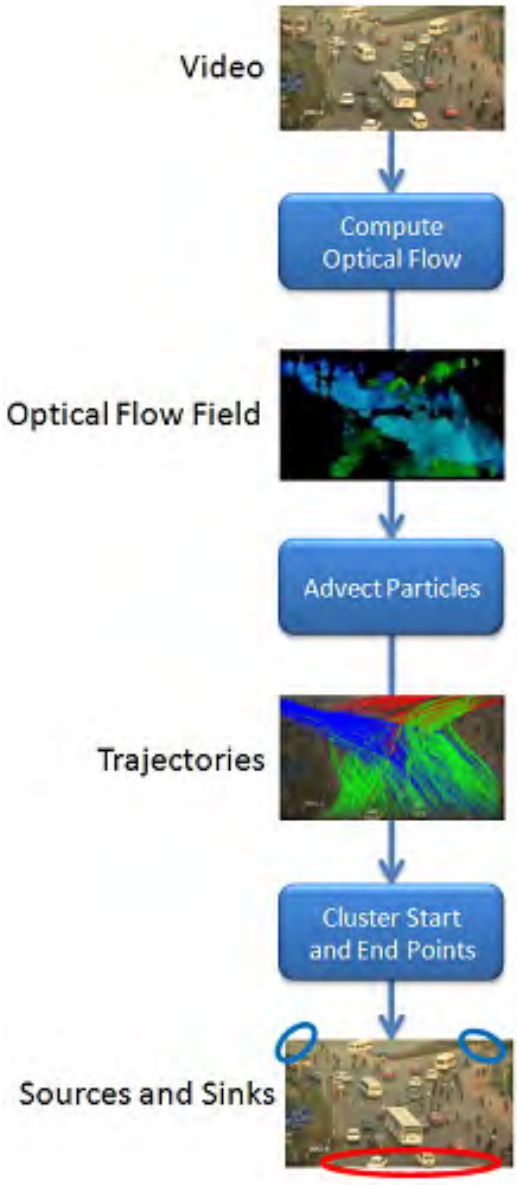


Figure 1.3: Flowchart of the proposed particle advection framework

describes the developed approach of advecting particles. The modeling of sources and sinks and a short overview over widely used clustering algorithms and proposed improvements to enhance the quality of clustering are described in Chapter 3. Chapter 4 and 5 show some experimental results of our particle advection approach and different clustering algorithms.

Chapter 2

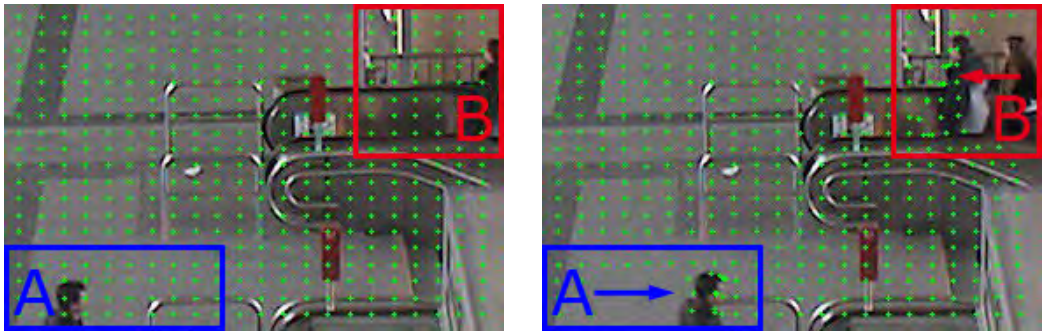
Real-Time Particle Advection

The basic idea of creating particle trajectories is to take video data (from file or webcam) as input, calculate and stack the optical flow fields, insert particles (Figure 2.1) and move them according to the corresponding motion vectors of the optical flow. Thus one can move the particles in a spatio-temporal environment. This Chapter describes the real-time particle advection algorithm and the challenges caused by analysis of real surveillance videos and the developed approaches.

2.1 Basic Algorithm

Given two consecutive frames of an image sequence, the real-time optical flow implementation calculates a motion vector $[u\ v]^T$ for every pixel position with subpixel accuracy.

Figure 2.1: $n \times m$ grid of particles at frame t_0



(a) time instant t_0 (b) time instant t_1

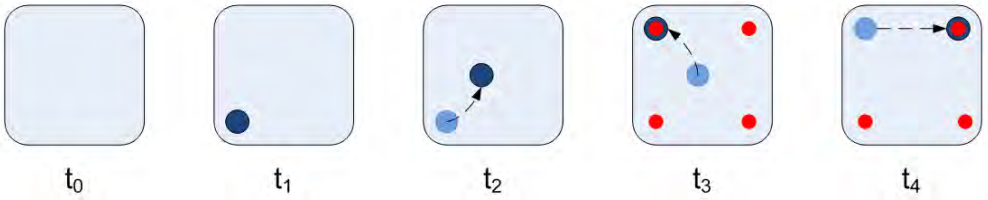
Figure 2.2: Forward advection of particles due to passenger movement

If a particle is dropped at the image position $[x_i \;\; y_i]^T$ at frame t_0, it will move to $[x_i + u \; y_i + v]^T$ in frame t_1 and so on. The trajectory of the particle is given by its entire coordinates over time. The basic algorithm is depicted in Algorithm 1.

Algorithm 1 Basic algorithm

```
1: t ← 0
2: insert new particles
3: grab video frame f_t
4: while video stream is available do
5:    grab next video frame f_{t+1}
6:    calculate motion vectors for every pixel between actual video frame f_t and the
      following video frame f_{t+1}
7:    for all particles do
8:       move particle according to its corresponding motion vector
9:       if particle gets stranded or leaves the video frame then
10:         remove particle
11:      end if
12:   end for
13:   if t is a multiple of t_insert then
14:      insert new particles
15:      move particles backward
16:   end if
17:   t ← t + 1
18: end while
```

Figure 2.2 shows two time instants of a video where three people use the escalators. Person A (at the bottom of the video frame, marked by the blue rectangle) is moving towards an escalator, Person B and C are located at the top of the video frame (marked by the red rectangle), using another escalator. Particles are moved from left to right due to the movement of Person A at the bottom. Particles are also moved from right to left due to the movement of Person B and C at the top of the video frame shown in Figure 2.2b.



(a) Backward advection

(b) Trajectories without backward advection

(c) Trajectories with backward advection

Figure 2.3: Backward advection and resulting trajectories

Definition 2.1. Active particles are particles which are still moving and not being stranded.

As particles can get stranded and thus being removed, new particles are inserted on a regular basis. All newly inserted particles are advected backwards, before they are advected forward to obtain longer and better trajectories. Why is this important? Figure 2.3a shows an example clarifying the importance of backward advection. Assuming there was no motion for longer than t_{strand}, so all particles stranded (t_0). Let the next particle insertion be at t_3. At t_1 a new object (dark blue ball) appears at the lower left corner, moves to the center (t_2), and afterwards to the upper left corner (t_3). At t_3, new particles (red) are inserted into the system. As the ball moves from the upper left to the upper right corner (t_4), the particle follows this motion and moves together with the ball.

If the particle were not advected backwards, the only identified movement would be from the upper left to the upper right corner, leading to the trajectory depicted in Figure 2.3b. Analyzing this trajectory would suppose the ball appeared at the upper left corner first and moved straight to the upper right corner. However, the ball appeared at the lower left corner first. To overcome this problem, all newly inserted particles are advected backwards. Hence the upper left particle depicted in Figure 2.3a at t_3 will go back to the center where the ball was at t_2 and also back to the lower left corner, where the ball was inserted (t_1). After this step, the particle advects forward from the upper left corner (t_3) to the upper right corner (t_4). The resulting trajectory is the combined backward and forward advection and leads to the correct trajectory depicted in Figure 2.3c.

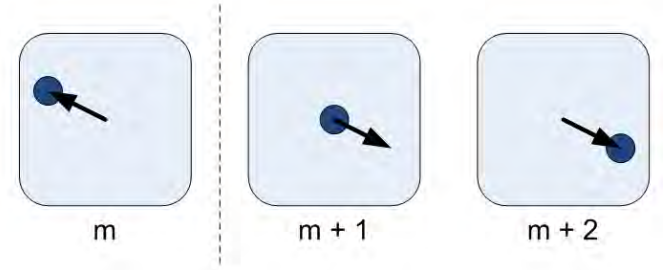
During backward advection, particles can also get stranded as during forward advection – this reduces the probability of particle hopping. We have implemented two different ways for calculating the motion vectors for backward advection:

- **Approximate Backward Advection:** The first method is an approximation and assumes that the motion of the particle is very smooth (which is often the case) and that it is very unlikely that it changes direction abruptly. Therefore it is sufficient to estimate the motion in the preceding frame as the inverted current motion vector by getting the motion vector $[u\ v]^T$ at the current particle position $[x\ y]^T$ in frame $m+1$. The estimated particle position in frame m would be the current particle position $[x\ y]^T$ minus the motion vector $[u\ v]^T$ depicted in Figure 2.4a.
- **Exact Backward Advection:** The second approach provides the exact motion vectors by finding the best matching motion vector within an $i \times i$ window around the current particle position (dotted rectangle at frame m in Figure 2.4b). Therefore the Euclidean distances between all motion vectors and the current particle position are compared – the motion vector with the minimal Euclidean distance is taken as the best matching motion vector. This approach finds the motion vector in frame m, which exactly points at the current particle position $[x\ y]^T$ in frame $m+1$, but it is much more time consuming, due to the consideration of all motion vectors within the $i \times i$ window.

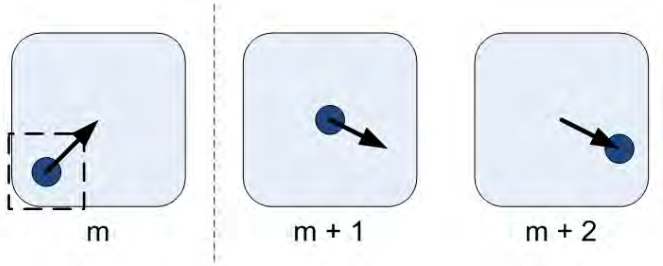
Figure 2.4 compares these two approaches – the negation of the motion vector of frame $m+1$ in Figure 2.4a supposes the ball to be in the upper left corner in frame m, though the exact motion vector would be in the lower left corner, depicted in Figure 2.4b. Usually both methods should provide similar results and the deviation should not be very big because of short and smooth motions, but the first method is only an approximation and may lead to different/wrong results.

Figure 2.5 shows the function of the particle advection algorithm over time – at the beginning t_0, particles are inserted into the system (blue arrow). If a particle only moves within a certain radius r over a particular time t_{strand} or leaves the video frame, it is stranded; hence it is removed from the system. This is required to make sure that only active particles are in the system, thus avoiding particle hopping.

After a fixed time t_{insert} new particles are inserted either at all initial $n \times m$ grid positions ($n \cdot m$ particles are inserted, Figure 2.6a) or at randomly chosen k initial grid positions illustrated in Figure 2.6b. The maximum of k (number of particles to be inserted) is restricted by the number of available grid positions ($k \leq n \cdot m$) as the



(a) Approximate motion vector determined through negation



(b) Exact motion vector determined through best matching

Figure 2.4: Comparison of motion vectors

number of inserted particles cannot be higher than the number of grid positions. One could insert particles only at positions having a high amount of motion, but this would exclude areas in advance, even though motion may occur unexpectedly. If there is no motion, the particles will be removed eventually, because they strand anyway. The newly inserted particles are first advected backwards for a particular time t_{back} (green arrow in Figure 2.5), and are subsequently advected forward afterwards.

2.2 Implementation

A particle set is inserted in regular time intervals t_{insert}, and particles must be removed when they have stranded. Hence many insertions and deletions of particles occur; therefore the implementation requires an efficient data structure. An efficient way to handle

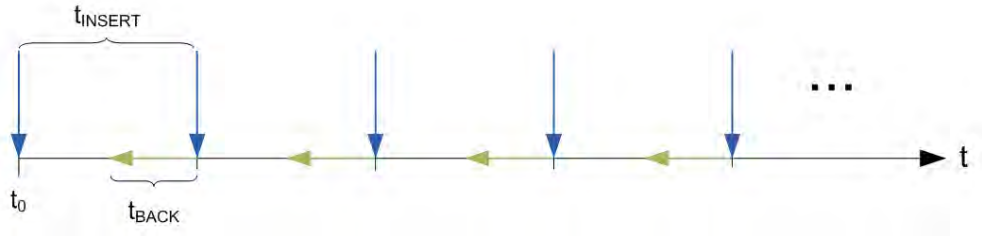


Figure 2.5: Timeline

(a) particles at all 9 × 7 grid positions

(b) 40 randomly chosen particles within 9 × 7 grid positions

Figure 2.6: Insertion of new particles

this are linked lists, where the particles are organized in a (single) linked particle list, depicted in Figure 2.7. This list is created during the initialization process. Therefore the list is created and the particles are inserted into an $n \times m$ grid, which drops the particles uniformly distributed over the frame. Each element in this list represents exactly one particle with its current coordinates and a list of previous coordinates. It also contains a "next" pointer to the next element of the list and only contains active particles. If a particle is marked as stranded, it is removed and the particle respectively its coordinates are stored in a file – if the particle actually moved. If the particle was inserted and stranded immediately, it will not be stored because particles without movement are not relevant for motion analysis.

The advantage of lists is the efficiency of sequential insertion and deletion processes, since only memory has to be allocated / freed and links between elements have to be changed. Another advantage of lists is dynamic memory allocation, facilitating the dynamical insertion of new particles. New particles are always added at the end of the list, therefore only new memory has to be allocated and the next pointer has to be changed to point to the new element. Particles are removed during forward advection if they get stranded or leave the video frame. Therefore the next pointer from the element before the element that should be removed has to be changed to point to the element after the element that should be removed and memory has to be freed. Figure 2.8 shows an example, where particle j is removed from the particle list.

Figure 2.7: List of particles

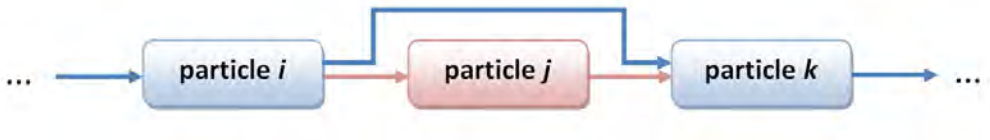


Figure 2.8: Removing particle j (red) from particle list

To obtain a contiguous trajectory, all coordinates for every particle are saved in a doubly linked list (a doubly linked list is required because of Backward Advection). Figure 2.9 shows the coordinates (orange) at given timestamps t_i for n particles. If a particle is moved according to the motion vector, a new coordinate is inserted at the end of the list. Coordinates are only added (and not removed) – therefore the structure of a linked list is reasonable and very efficient.

If a particle is marked as stranded, it is removed from the particle list and the coordinates are written to an output file. Video footage often contains areas where little or no movement at all occurs – so there are many particles which do not move at all and are stranded just after the insertion process. These particle coordinates are not relevant, because they do not contain information about motion. Thus they are not saved to file to reduce the overall amount of particles and file size.

The particle advection algorithm is implemented in C using the open source library OpenCV and FlowLib [14] which calculates the optical flow on the GPU (graphics processing unit) efficiently. The dense optical flow implementation calculates the motion vectors in real-time for every frame. Hence, motion vectors are available for exactly one frame and they are used to move particles forward. To advect particles backward in time

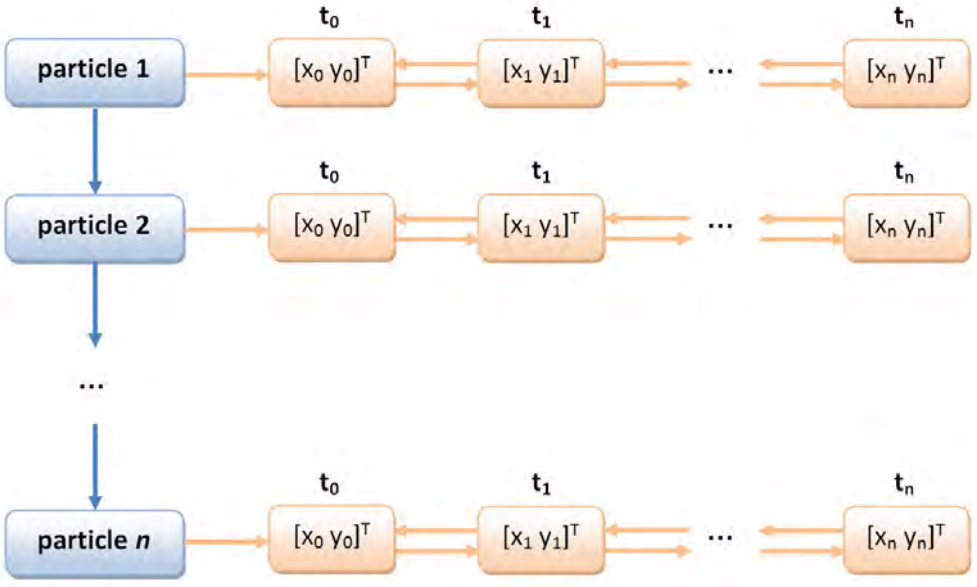


Figure 2.9: Coordinate list (orange) for particles (blue)

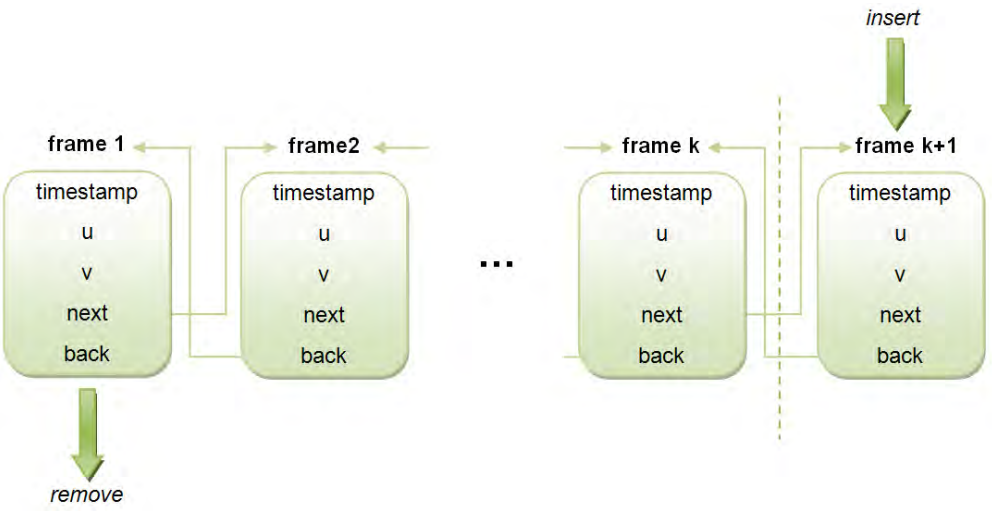


Figure 2.10: Dynamic insertion/deletion of motion vectors

the motion vectors of the last k frames must be known – k represents the number of frames passed by during time t_{back}. The first idea might be to save all motion vectors as soon as they are calculated, but this approach is only feasible for a limited video sequence. Not all motion vectors are needed as the backward advection is restricted on the last k frames, hence only the last k motion vectors are stored as k elements in a list, where k acts as buffer size. All calculated motion vectors are added at the end of the list. As long as the current frame number is lower than k, the motion vectors are only added to the list. If the current frame number equals $k+1$, the current motion vectors are added and the first element (oldest motion vectors) is removed. Thus only the last k elements (motion vectors at the last k frames respectively during time period t_{back}) are available, illustrated in Figure 2.10.

2.3 Challenges

Due to analysis of real surveillance videos, unforeseeable influences like changing weather or lighting conditions are hard to handle and have major effects on the quality of the particle advection process. But also the camera position as well as occlusions affect the quality enormously. This Section describes two major effects influencing the quality of trajectories.

2.3.1 Occlusions

Particles are advected according to optical flow calculation, not considering the structure of a scene or any semantic knowledge. Due to occlusions, particles might not follow the

(a) Occlusions due to street lamps (b) Occlusions of moving objects

Figure 2.11: Types of occlusions influencing particle advection

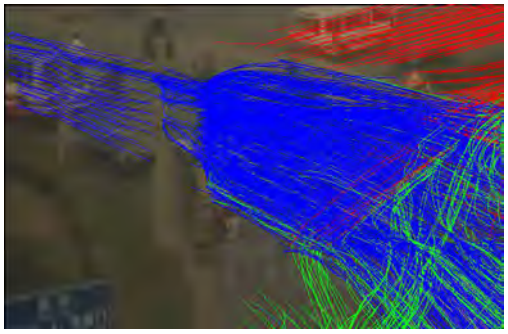
correct motion flow. Instead, there are two possibilities occlusions can lead to:

1. **Static Occlusions:** a particle does not move any more, because the object (or part of the object) moves behind a static object (e.g. a car is occluded by a street lamp, illustrated in Figure 2.11a). If a moving object with particles attached to it is occluded by a static object, the optical flow algorithm cannot detect the correct object motion. Hence, the particle is not attached to the moving object any longer and does not move any more. Even if the moving object appears after the stationary object again, the particles attached to the moving object are still at the position where the occlusion began. Figure 2.12b illustrates the problem: particles are not attached to the bus any more but got stuck at a street lamp. As static objects do not move in general, particles do not move either and are removed due to the stranded criterion. Figure 2.12c shows interrupted trajectories, thus resulting in wrongly detected sinks, because many particles strand at positions where occlusions take place. Not only sinks are wrongly detected due to occlusions but also sources may be detected wrongly while using backward advection. This is due to the fact that particles cannot pass the obstacle in either way. Figure 2.12d depicts longer trajectories by passing the obstacles.

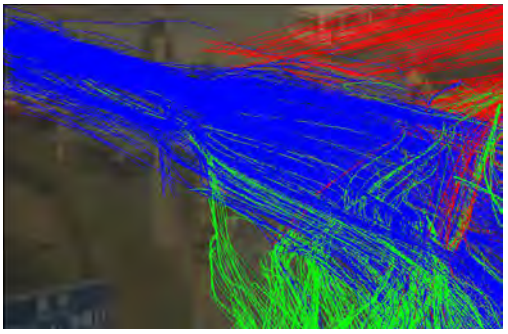
2. **Dynamic Occlusions:** a moving object is occluded by another moving object and the particle jumps from one object to the other and follows another object after occlusion shown in Figure 2.11b. As we are interested in motion flows and not in tracking individuals, particle hopping from one moving object to another moving object heading the same direction need not be detected as particle hopping. Hence only particle hopping from objects heading in "different" directions and thus belonging to another motion flow, should be detected. Failing to detect particle hopping would result in wrong trajectories because the trajectory of one particle is a mixture of trajectories caused by different objects moving in opposite directions.

(a) Occlusions due to street lamps

(b) Particles get stuck due to occlusions



(c) Interrupted trajectories due to occlusion



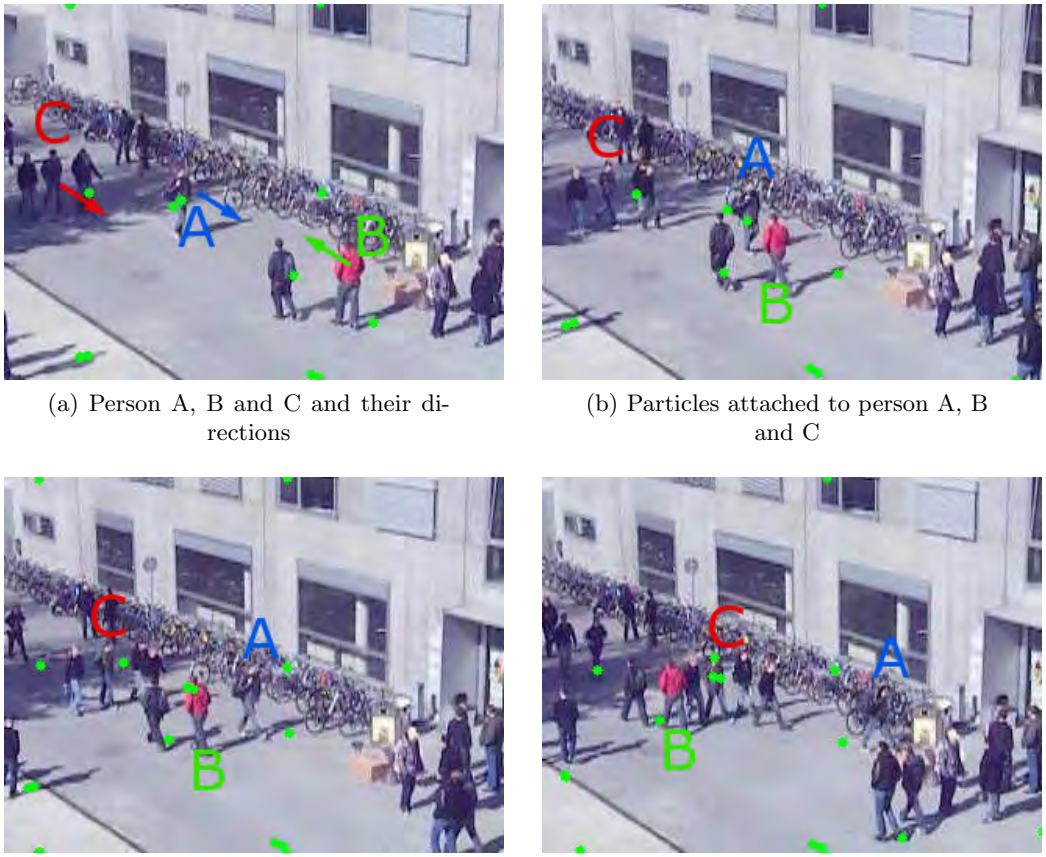
(d) Longer trajectories due to passing obstacles

Figure 2.12: Influence of static occlusions on particle advection

Figure 2.13 shows an example of an image sequence, where particle hopping adulterates the trajectory. In this example, particle hopping occurred twice – at the beginning the particle followed person A going from left to right. Person B (heading from right to left) crossed person A's way and the particles jump from person A to B and follow person B depicted in Figure 2.13b and 2.13c. Figure 2.13c and 2.13d show another person C, also heading from left to right crossing person B's way and the particles jump from person B to person C and follow person C. In this example, person A and C are heading in the same directions. If the particle had jumped only once, it would head in the opposite direction than person A is heading to. In general one can say that particle hopping results in unpredictable falsification of trajectories, therefore a mechanism to detect particle hopping is needed. Assuming that motion of objects is smooth leads to the developed particle hopping detection mechanisms.

2.3.2 Perspective View

Objects in the front near to the camera appear bigger than objects further away from the camera, depicted in Figure 2.15a. Hence small movements in the front can be detected



(a) Person A, B and C and their directions

(b) Particles attached to person A, B and C

(c) Particles hopped from person A to B

(d) Particles hopped from person B to C

Figure 2.13: Particle hopping

more easily than small movements in the back as the resolution decreases depending on the camera distance. A movement of one pixel in the foreground can be equal to a movement of one cm in world coordinates whereas a one-pixel movement in the back can be equal to a movement of one meter in world coordinates.

Finding plausible measurement units for the stranded criterion defined in Definition 1.5

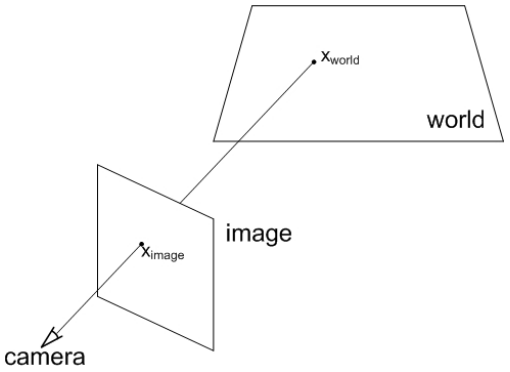
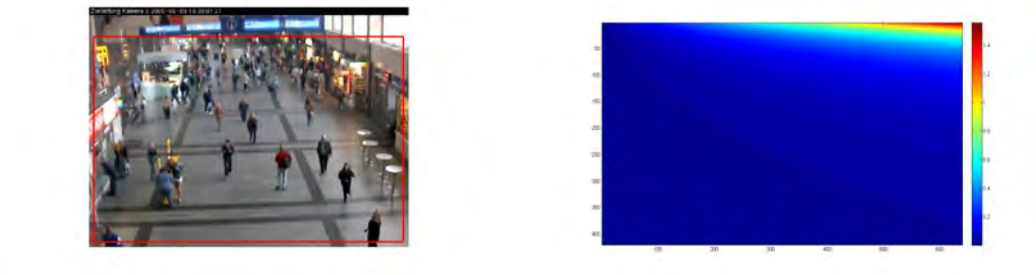


Figure 2.14: Relation between camera, image plane and world plane



(a) Perspective view of a train station

(b) Relation between image coordinates and world coordinates

Figure 2.15: Perspective view and homography

is challenging: if the radius is measured in pixels, the real movement will be much smaller in the front than in the back when using the same radius for the whole scene. The knowledge of (camera) calibration parameters helps to apply the stranded radius equally to the front as well as to the back of the scene as follows:

$$[x_{\text{world}}\ y_{\text{world}}\ 1]^T = H^{-1} \cdot [x_{\text{pixel}}\ y_{\text{pixel}}\ 1]^T, \tag{2.1}$$

where H is a 3×3 matrix, transforming pixel coordinates p_{image} into world coordinates p_{world}. Homography matrix H [18] describes the relation between the pixel and a world plane, depicted in Figure 2.14. Applying the inverse homography matrix H^{-1} on the pixel coordinates leads to world coordinates.

Figure 2.15b shows the relation between a change of one pixel in y direction in image coordinates and the change of x direction in world coordinates. The dependency between pixel and world coordinates was calculated within the red boundary depicted in Figure 2.15a. A change of 1 pixel in image coordinates in y direction results in a minimum and maximum movement of 3.65 mm and 1.56 m in x direction and a minimum and maximum movement of 1.05 mm and 0.45 m in y direction. The maximum of movement is coded red in the upper right corner of Figure 2.15b.

Hence not only pixel coordinates are applied when using the stranded criterion, but also world coordinates can be used. This leads to a more stable (depending on the accuracy of the estimation of H) approach as the stranded criterion in the front is the same as it is in the back, requiring the knowledge of calibration parameters or the inverse homography matrix.

Due to perspective distortions and restricted resolution the use of world coordinates instead of image coordinates can improve results, but it does not remove the uncertainty in different directions. World coordinates can be recalculated using camera calibration information, but the camera resolution cannot be changed. Thus the change of 1 pixel

in one direction (e.g. y direction) may yield in a higher change in world coordinates than the change in another direction (e.g. x direction). Due to this fact, the use of a circle as stranded criterion may be inefficient as the use of an ellipse may provide better results as it is able to fit the diverse uncertainties much better.

2.4 Improved Algorithm

Algorithm 2 Improved algorithm

```
1: t ← 0
2: insert new particles
3: grab video frame f_t
4: while video stream is available do
5:    grab next video frame f_{t+1}
6:    calculate motion vectors for every pixel between actual video frame f_t and the
      following video frame f_{t+1}
7:    for all particles do
8:       if particle hopping detected then
9:          remove particle
10:      else
11:         move particle according to its corresponding motion vector
12:         if particle leaves the video frame then
13:            remove particle
14:         else if particle gets stranded then
15:            delete coordinates of the last k frames according to t_strand
16:            advect particle on reduced image resolution for the last k frames
17:            if particle strands as well then
18:               remove particle
19:            end if
20:         end if
21:      end if
22:   end for
23:   if t is a multiple of t_insert then
24:      insert new particles
25:      move particles μ frames backward
26:   end if
27:   t ← t + 1
28: end while
```

The challenges described in Section 2.3 result in an improved particle advection algorithm, shown in Algorithm 2. The following describes the approaches in detail:

- **Coping with static occlusions:** To avoid particles getting stuck at obstacles, a hierarchical approach is used, which facilitates particles to pass small obstacles. An exact definition of "small" obstacles is desirable, but "small" cannot be specified as an obstacle being smaller than e.g. two pixels as the current amount of motion

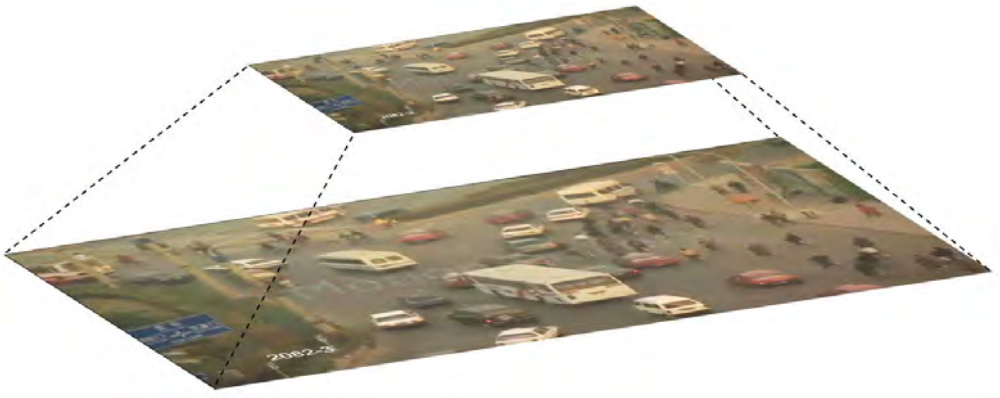


Figure 2.16: Image pyramid showing original and reduced video size

has to be taken into account – the higher the amount of motion, the "larger" the obstacle can be to be passed by this approach. Therefore, particle advection does not take place on full video resolution, but rather on a reduced video size where obstacles appear much smaller. Thereby particles are more likely to pass obstacles because the amount of motion is higher than the size of obstacle resulting in less interrupted trajectories. Figure 2.16 shows the original and the reduced video resolution.

Every time a particle is detected as stranded, it is not removed immediately but advected on a higher image pyramid level (that means at a reduced resolution) for a certain time. If the particle strands as well, it is removed from the system. If it does not strand, the particle is advected for a certain time on the higher image pyramid level. Afterwards the particle advection is set back to the normal level and particle advection is continued at the original video size.

Using this approach means accepting some inaccuracies while advecting particles. Higher level advection results in longer trajectories (Figure 2.12c compared to Figure 2.12d) thus facilitating the clustering process for finding sources and sinks, described in Chapter 3. Obviously this approach only works well for small obstacles like street lamps and does not provide a reasonable solution for objects like billboards.

- **Coping with dynamic occlusions - Solution 1:** The first mechanism assumes that a moving object does not change its direction abruptly but smoothly. Before a particle is advected according to its corresponding motion vector provided by optical flow analysis, the average motion vector of the last l particle advection steps is calculated (for experimental results l has been set to 10). If the new motion vector is not "similar", an abrupt change of direction occurred and it is very likely that particle hopping took place. A way of measuring the similarity

(a) Valid motion vector if strict particle hopping criterion is applied ($\lambda_t = 45°$)

(b) Invalid motion vector if strict particle hopping criterion is applied ($\lambda_t = 45°$)

(c) Valid motion vector if strict particle hopping criterion is applied ($\lambda_t = 90°$)

(d) Invalid motion vector if strict particle hopping criterion is applied ($\lambda_t = 90°$)

Figure 2.17: Particle hopping criterion

between motion vectors is to calculate the angle λ between them, defined as:

$$\lambda = \arccos \frac{[u_{\text{avg}}\ v_{\text{avg}}]^T \cdot [u_{\text{new}}\ v_{\text{new}}]^T}{\left|[u_{\text{avg}}\ v_{\text{avg}}]^T\right| \cdot \left|[u_{\text{new}}\ v_{\text{new}}]^T\right|}, \tag{2.2}$$

where $[u_{\text{avg}}\ v_{\text{avg}}]^T$ is the average and $[u_{\text{new}}\ v_{\text{new}}]^T$ is the new motion vector. If λ is smaller than a specified threshold λ_t, the motion is identified as being smooth. Otherwise, particle hopping occurred and the particle is marked as "hopped". Figure 2.17 shows the last 10 motion vectors (blue), the average motion vector (orange) and a possible new motion vector (green or red, depending on the chosen threshold shown as black line). The choice of threshold influences the number of particle hops and decides whether abrupt changes of particle directions (e.g. 90 degrees) are caused by particle hopping or abrupt object movement. Particle hopping is only detected if the amount of motion is higher than a threshold.

- **Coping with dynamic occlusions - Solution 2:** The second mechanism compares the motion vector similarity by analyzing accelerations. Assuming motion to be smooth, acceleration of a particle should not change very much (except at the very beginning and end). If particle hopping occurs and the particle were advected in the opposite direction, the value of acceleration of this particle would be high. If the value of acceleration is above a threshold, the particle is marked as "hopped" as when using the first approach.

Chapter 3

Sources and Sinks

Start and end points of a perfect person's trajectory represent sources and sinks as they exactly define where the subject started and where it ended. Usually many particles generate trajectories, thus many start and end points are generated.

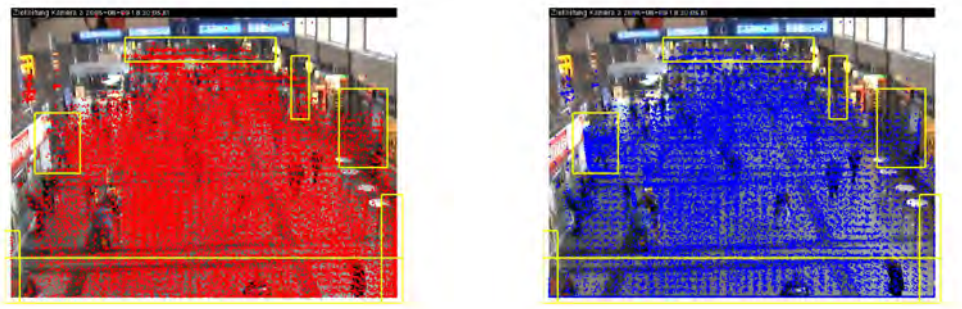
Definition 3.1. Valid sources and sinks are sources and sinks which actually exist in the real world or originated from occlusions.

Definition 3.2. Invalid sources and sinks are sources and sinks which do not exist in the real world. These often originate from broken trajectories.

Definition 3.3. Primary sources and sinks are sources and sinks which actually exist in the real world.

Definition 3.4. Secondary sources and sinks are valid from occlusions being not desired, but explainable.

Trajectory start and end points do not only represent valid sources and sinks. As described in Section 2.3, trajectories are not always perfect – hence start and end points may belong to invalid sources and sinks caused by errors (i.e. errors due to perspective distortions), respectively secondary sources and sinks caused by occlusions. Figure 3.1 shows typical results for start and end points of trajectories from a Viennese train station, not considering the density of these points. As one can see easily, obtaining primary sources and sinks by clustering is very challenging and further analysis needs to be done. Hence the calculation of the probability density function using a kernel density estimator ([19]) yields in easier interpretable results depicted in Figure 3.2. A kernel density estimator is an enhanced version of a histogram. To construct a histogram, the interval containing data points is divided into several subintervals, so called bins. For each bin, the number of data points within this represents the "density". As the



(a) Trajectory start points

(b) Trajectory end points

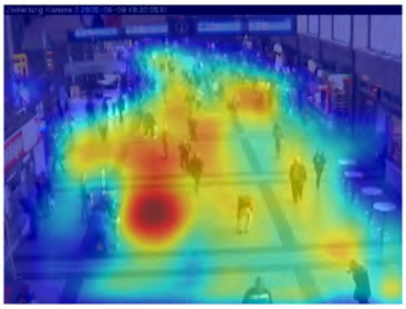
Figure 3.1: Plots showing start and end points and areas, where true sources and sinks should be found, marked by yellow rectangles

results do not deliver a smooth function and depend on the width for each bin, the kernel density estimator (KDE) was proposed. The KDE does not divide the interval into several bins, but uses different functions (e.g. Gaussian) to represent the density. Therefore a Gaussian kernel is placed over each data point - the sum of all Gaussians at all points is the desired density function.

Figure 3.2 shows the density function applied on two dimensional data. Colors used in this figure represent the density of start and end points - cold colors (e.g. blue) visualize small densities. The warmer the color gets (e.g. green, yellow, red) the more start and end points are placed in this area. This data is obtained from a video sequence showing a Viennese train station. The doors to different platforms are located on the right side of the picture, the escalator to the exit is located on the left side. In this video sequence a train arrives and passengers are leaving the platform located in the bottom right corner and are heading to the exit on the left side. Throughout the video sequence which lasts for 15 minutes, this was the only train arriving thus resulting in one main source and one main sink because most people directly left the train station. Figure 3.2b and Figure 3.2d depict results obtained by "optimal" particle advection settings - these settings use a long backward range, facilitating longer trajectories. The use of a short backward range results in shorter trajectories, thus making the finding of real sources more difficult. Figure 3.2a and 3.2c depict the results when using "suboptimal" parameters, where the true source was not found.

3.1 Modeling

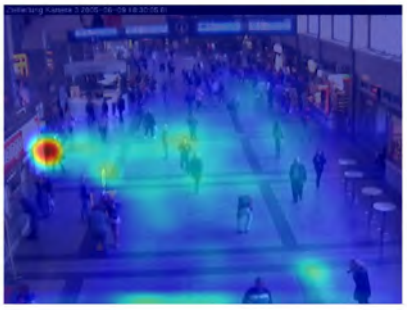
To obtain sources and sinks from start and end points of trajectories, clustering algorithms are applied. One source or sink does not necessarily need to be represented by



(a) Density plot of start points using suboptimal particle advection settings

(b) Density plot of start points using optimal particle advection settings

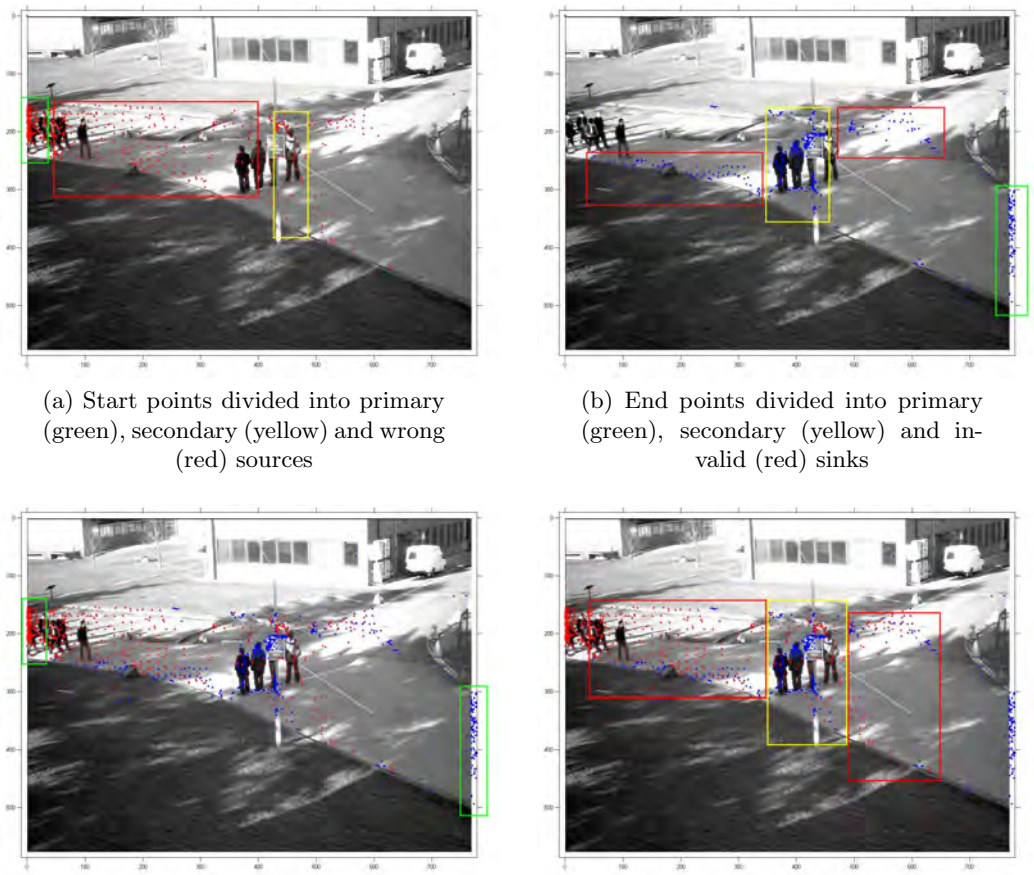
(c) Density plot of end points using suboptimal particle advection settings



(d) Density plot of end points using optimal particle advection settings

Figure 3.2: Influence of different particle advection settings on trajectory start and end points

exactly one cluster, because a large source or sink (e.g. entering or leaving the camera field of view) can be represented by many small clusters placed in the area of the big source or sink. To achieve reasonable clustering results the pair-wise similarity between data points is calculated by using the Euclidean distance, as all points are represented by 2 dimensional image coordinates. Invalid sources and sinks should be eliminated by the clustering algorithm, whereas secondary sources and sinks should by avoided by using better settings for the particle advection process (e.g. by using the developed approach against stranding particles beyond sinks, introduced in Section 2.3). Figure 3.3 depicts the differences between primary and secondary sources and sinks as well as invalid sources and sinks. Figure 3.3a shows all start points divided into primary sources (green), secondary sources caused by an obstacle (yellow) and invalid sources (red). End points and the corresponding primary sinks (green), secondary sinks caused by an obstacle (yellow) and invalid sinks are shown in Figure 3.3b. Figure 3.3c and 3.3d depict all start and end points divided into primary sources and sinks, secondary sources and sinks and invalid sources and sinks. These figures show that primary sources and sinks



(a) Start points divided into primary (green), secondary (yellow) and wrong (red) sources

(b) End points divided into primary (green), secondary (yellow) and invalid (red) sinks

(c) All points with primary (green) sources and sinks highlighted

(d) All points with secondary (yellow) and invalid (red) sources and sinks highlighted

Figure 3.3: Different types of sources and sinks

are only located at the left and right side of the picture, as these are the areas where subjects enter or exit the camera field of view.

3.2 Clustering Algorithms

This Section gives a brief introduction on different clustering algorithms. There is a huge variety of clustering algorithms, resulting from diverse clustering methods such as hierarchical or density-based methods. Some other important criteria to distinguish between different clustering methods are:

- **underlying data distribution:** Clustering algorithms can be distinguished by the need of the underlying data distribution to be specified or not. Usually the data distribution is not known, hence only assumptions are used thus influencing clustering results.

- **number of clusters:** Many clustering algorithms use the expected number of clusters as input argument, others estimate the number of clusters automatically.

Definition 3.5. A **cluster** is a subsample of data points, having small distances between these points.

To obtain distinguished clusters, the distance between different clusters should be high [1]. Only the basic principle of a few frequently used clustering algorithms will be explained in short. All introduced algorithms are unsupervised, meaning that no labeled training data is available and the clusters have to be found automatically by the algorithm.

3.2.1 K-Means

The inputs of this algorithm are n dimensional, unlabeled data points and an expected number k of clusters. Each cluster is represented by one of k vectors, which describes the cluster center. To find the local optimum of these cluster centers, an iterative process enhances the quality of clustering.

Algorithm 3 K-Means

Require: number of cluster centers k
1: randomly locate cluster centers
2: **repeat**
3: **for all** data points **do**
4: calculate Euclidean distance to all cluster centers
5: data point belongs to the cluster center having the smallest distance
6: **end for**
7: move cluster center to the mean of all data points assigned to this cluster
8: **until** cluster assignments do not change or maximum number of iterations reached
9: **return** data points and their corresponding cluster

The basic algorithm is described in Algorithm 3. At first, cluster centers have to be initialized – this is often done randomly. Figure 3.4a shows the unlabeled data points (green) and the initialized cluster centers (red and blue). In our example depicted in Figure 3.4, the number of clusters k has been set to two. In the next step, the Euclidean distances between each data point and all cluster centers are calculated. Each point is now assigned to the cluster having the minimum cluster center distance. The results of this step are two clusters containing all data points, separated by the magenta line and shown in Figure 3.4b. These clusters are not intuitive, as humans would have clustered the data points differently. To correct the clusters, the cluster centers are set to the mean value of all data points assigned to the corresponding cluster depicted in Figure 3.4c.

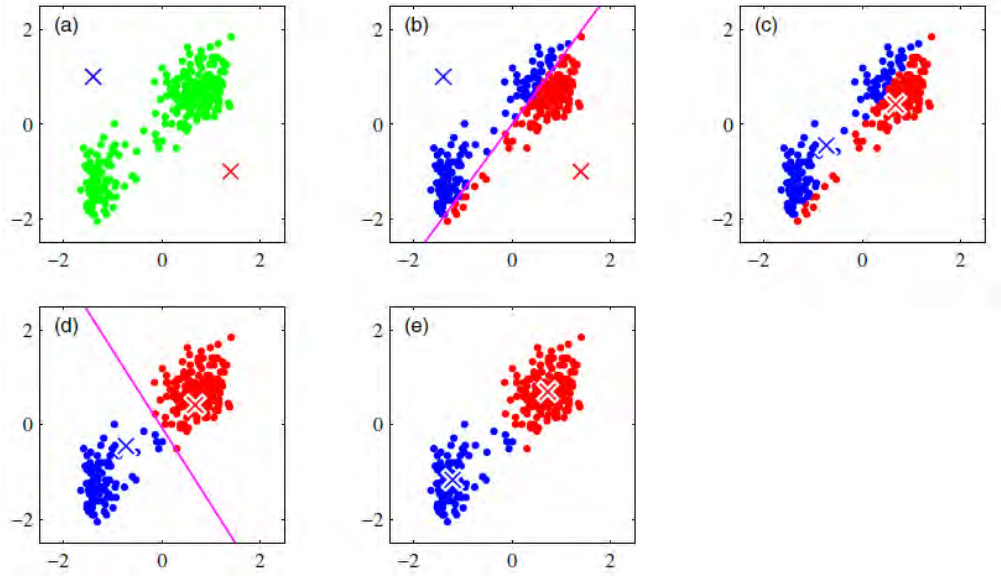


Figure 3.4: k-means (first two iterations) [1]

These two steps (assignment of data points to the closest cluster center and updating the cluster centers) are processed repeatedly, until the algorithm converges (cluster assignments do not change or a maximum number of iterations is accomplished). Figure 3.4d shows the second assignment of data points to their nearest cluster center; Figure 3.4e depicts the change in cluster centers according to the new mean value of data points.

This simple approach has the drawback of only getting local optima. Therefore, different initializations of cluster centers yield in different results, which is not desired. As the input data is unlabeled, the correct estimation of k is very difficult, hence an automatic selection of cluster numbers is preferable.

3.2.2 Expectation Maximization

Definition 3.6. A **normal distribution**, also called **Gaussian distribution** is a continuous probability distribution and is defined as:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \cdot e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}, \tag{3.1}$$

where μ is called mean and σ is called variance.

Assignment of data points to clusters can be done hard or softly, meaning that hard assignments assign each data point to exactly one cluster (i.e. k-means) and soft assignments only assign a membership probability. Using Expectation Maximization, the distribution of data points is assumed to be a mixture of normal distributions, which

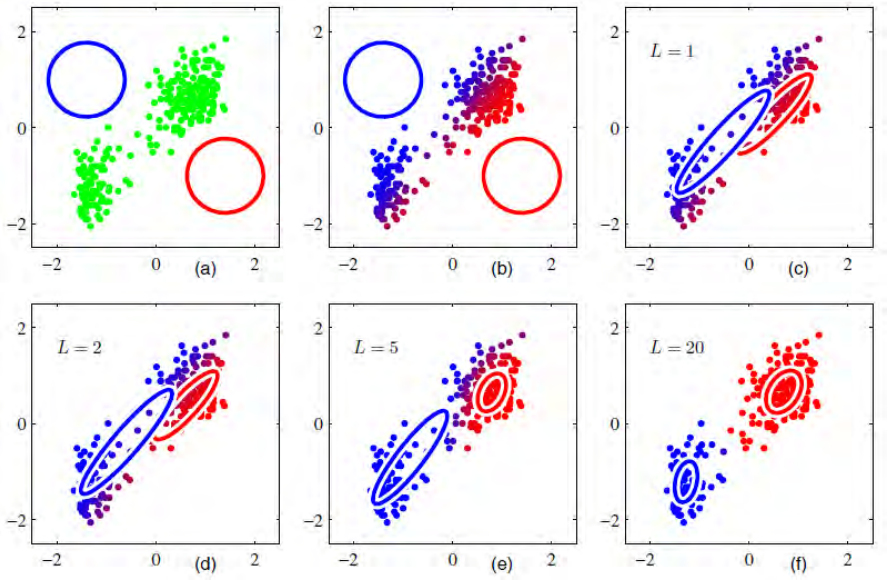


Figure 3.5: Expectation maximization [1]

yields in more flexible results than k-means, as the underlying data structure is taken into account (although assuming the underlying distribution to be a mixture of Gaussians). Given that each cluster is generated by a normal distribution, the clustering algorithm has to find the mixture of normal distributions and estimate their parameters. Every data point will not be assigned to one cluster but gets a probability of membership to every Gaussian distribution.

K-means is very similar to Expectation Maximization, but not assuming the underlying distribution to be a mixture of Gaussians. For the sake of completeness it should be mentioned that Expectation Maximization can be used assuming arbitrary distributions and not only mixtures of Gaussians.

3.2.2.1 Basic Algorithm

The initialization of the Expectation Maximization algorithm is similar to the initialization of k-means (Section 3.2.1): it is often done randomly, but if the initialization fits the underlying data, then the algorithm will converge faster. As the number of iterations to gain an applicable estimation of distribution parameters (means, covariances) is much higher and more computations are necessary than using k-means, the initialization step is much more important. Hence the results of k-means are often used as initialization for the Expectation Maximization [1]. Figure 3.5a shows the unlabeled data points (green) and the same initialization for the means of the Gaussian distributions (red and blue) as in k-means (where the means were called cluster centers).

Algorithm 4 Expectation Maximization

Require: number of cluster centers k
1: **for all** clusters **do**
2: randomly choose mean μ and covariance σ
3: **end for**
4: **repeat**
5: **for all** data points **do**
6: assign cluster membership probabilities to each data point ("Expectation")
7: re-estimate Gaussian distribution parameters μ and σ ("Maximization")
8: **end for**
9: **until** changes of parameter being lower than a threshold or maximum number of iterations reached
10: **return** data points and their corresponding cluster

Expectation Maximization's basic algorithm is shown in Algorithm 4. The expectation step uses the current distribution parameters of the clusters to assign membership probabilities to every data point. Data points having a high probability for belonging to a cluster are either marked as red or blue in Figure 3.5b, depending on the cluster they belong to (red or blue). Points that cannot be clearly assigned to one of the clusters are marked by a mixture of red and blue depending on their respective membership probabilities, i.e. data points having equal membership probabilities to the red and blue distribution are marked purple (50% red and 50% blue). The re-estimation of Gaussian distribution parameters occurs in the second step, the maximization step. Due to assigned probabilities, the means and covariances of the Gaussian distributions are re-estimated considering the current membership probabilities.

These two steps are processed repeatedly to increase the accumulated membership probabilities. Because of the maximization step, the Gaussian distributions are going to fit the underlying data points better and better. Figure 3.5c depicts the Gaussian distributions after one iteration, Figure 3.5d after two iterations, Figure 3.5e after five iterations and finally, the enhanced clustering after 20 iterations is shown in Figure 3.5f. This algorithm terminates when the changes in parameters are below a specified threshold or the number of iterations exceeds the maximum number of iterations. Furthermore it is not as fast and simple as k-means, but leads to more accurate results as the underlying data distribution can be taken into account.

3.2.2.2 Improvements

When analyzing real surveillance videos, clustering trajectory start and end points into sources and sinks is very hard because of imperfect trajectories caused by noise, tracking errors or changing lighting conditions. Hence, many of the start and end points

of the extracted trajectories do not correspond to real sources and sinks and have to be removed. We assume that wrongly detected sources and sinks are represented by widely spread Gaussian distributions, whereas correctly detected sources and sinks are represented by dense Gaussian distributions [15]. This assumption is also motivated by the observation that the density of start and end points is significantly higher in regions around real sources and sinks, respectively, since noise, tracking errors and changing lighting conditions have only a minor effect on the start and end points distribution over time.

This observation can be exploited to improve the results, as wrong source clusters can be removed by introducing a density criterion. The measure of density d_i of the ith cluster is defined as

$$d_i = \frac{w_i}{\pi \cdot \sqrt{\|\Sigma_i\|}}, \tag{3.2}$$

where w_i is the prior probability and Σ_i the covariance matrix of the ith cluster. If the density of the ith cluster is lower than the specified threshold T, the ith cluster will be classified as noise cluster [15]. Threshold T is defined as:

$$T = \frac{\alpha}{\pi \cdot \sqrt{\|\Sigma\|}}, \tag{3.3}$$

with α being a user defined weight ($0 < \alpha < 1$) and Σ the covariance matrix of the entire data set. The norm used in Equations 3.2 and 3.3 is the Frobenius norm.

Another improvement proposed by [20] is called PG-means (projected-Gaussian means) which detects the number of clusters k automatically. Starting with one cluster, k is increased by an iterative process if the current model does not fit the underlying data. Therefore the algorithm learns the model by using the Expectation Maximization (EM) algorithm. If EM converges, the data and the model are projected into one dimension. This can be done because a projection of a Gaussian mixture remains a Gaussian mixture while using linear projections. Executing a model fitness test is much easier to be done in one dimension than in higher dimensions. The fitness of the projected model to projected data is tested by using the *Kolmogorov-Smirnov* test [21]. Multiple executions of this test lead to multiple results – only if all tests reveal that the current model fits the data, the algorithm terminates and the number of clusters is found. Otherwise, k is increased by one and a new model has to be found.

3.2.3 Mean Shift

Mean shift [22] does not make any assumptions about the underlying data distribution or number of clusters as it is needed for e.g. expectation maximization. The n-dimensional data points are a sample generated by probability density functions and mean shift finds the modes of these functions. The number of modes is equal to the number of cluster centers k.

Algorithm 5 Mean Shift

Require: bandwidth b
1: $i \leftarrow 0$
2: **while** some data points do not belong to any cluster **do**
3: $i \leftarrow i + 1$
4: choose data point randomly from data points not belonging to any cluster
5: use this point as cluster center
6: **repeat**
7: **for all** data points **do**
8: calculate distance d to start point
9: **if** $d \leq b$ **then**
10: add vote to current data point for cluster i
11: **end if**
12: **end for**
13: move cluster center towards the maximum increase of density
14: **until** movement of cluster center is lower than a threshold
15: if possible, merge cluster centers
16: **end while**
17: data point belongs to the cluster having the highest number of votes
18: **return** data points and their corresponding cluster

Algorithm 5 shows the basic function of the algorithm. Starting with a randomly chosen data point, a density estimator is applied, estimating the density of data points within a specified radius called bandwidth – depicted by a circle in Figure 3.6. The points within this circle are marked as belonging to the same cluster as the chosen data point. Afterwards the gradient of this density function is calculated resulting in the *mean shift vector* pointing towards the direction of maximum increase of density. The preliminary cluster center is now moved into the direction of the mean shift vector. These steps are executed repeatedly until the movement of the cluster center falls below a threshold. Successive computations result in a path of the cluster center to the maximum of density shown in Figure 3.6. If this movement converges, the algorithm starts again with a randomly chosen data point which does not belong to any cluster, the number of clusters is increased by one and the algorithm starts again. This will be done until all data points belong to a cluster.

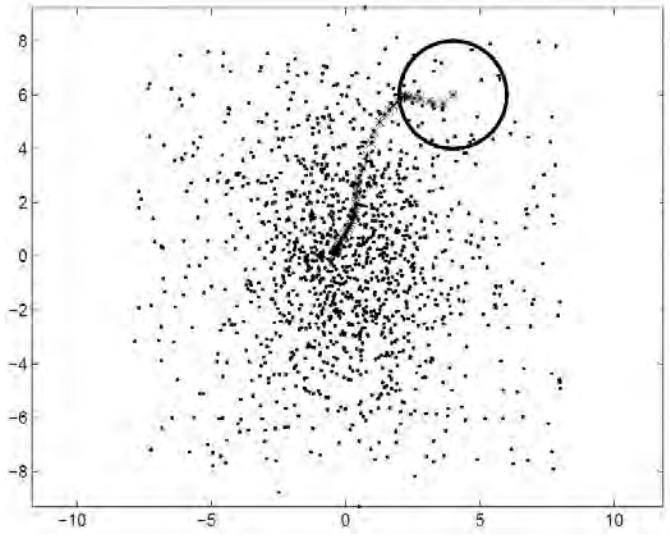


Figure 3.6: Successive computations of mean shift [2]

If two cluster centers are close to each other, they are merged. To resolve conflicts of one data point belonging to more than one cluster, each data point gets votes of belonging to one cluster. After the convergence of this algorithm, every data point belongs to the cluster having the highest number of votes.

3.2.4 DBSCAN

This algorithm is based on the Gestalt law of proximity: humans interpret near objects to belong together, whereas sparse objects are not belonging together [23]. Figure 3.7 depicts 9 equal dots, but having different distances. Humans interpret these 9 equal dots as 3 groups of dots, containing 4, 3 and 2 dots. [3] developed a new clustering algorithm based on a density based notion called DBSCAN. This Density-Based algorithm was designed for large spatial databases containing noise and discovers clusters of arbitrary shape without the need of specifying the number of clusters in advance making the assumption that the density of points within clusters is much higher than the density between clusters or areas of noise.

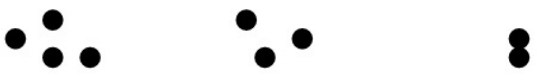


Figure 3.7: Gestalt law of proximity

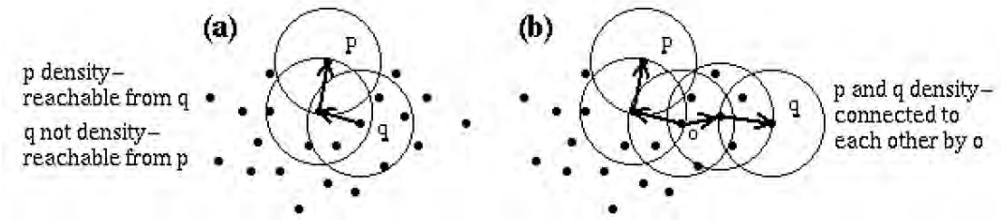


Figure 3.8: DBSCAN: (a) density-reachable, (b) density-connected [3]

3.2.4.1 Basic Algorithm

The fundamental idea of DBSCAN is the neighborhood analysis of a given point p_0 within a radius ϵ. If the number of points within this radius is higher than a specified threshold *MinPts*, every point p_i is called directly density-reachable. If a point q is not directly density-reachable from p, but there is a chain of points whereby q gets reachable, this point is called density-reachable from p. Given that q is not density-reachable from p, but a point o exists, from which both are density-reachable, p and q are called density-connected (Figure 3.8).

A cluster is defined as the maximal subset of points, which are at least density-connected; all other points are classified as noise or belong to other clusters. In short: all points, which are density-reachable or density-connected to a specific point p belong to the same cluster as p, as long as the number of these points is higher than the threshold *MinPts*. The basic algorithm is depicted in Algorithm 6.

Algorithm 6 DBSCAN

Require: minimal number of objects considered as a cluster $MinPts$
1: $i \leftarrow 0$
2: calculate radius ϵ
3: **repeat**
4: randomly choose a data point p
5: calculate Neighborhood N_p {find all density-reachable and density connected data points to p}
6: **if** number of points in $N_p \geq n$ **then**
7: all points in N_p belong to a cluster
8: $i \leftarrow i + 1$
9: **end if**
10: **until** all data points belong to a cluster
11: **return** data points and their corresponding cluster

3.2.4.2 Improvements

Because of using real surveillance videos, the same density criteria as introduced in Section 3.2.2.2 will be used during the experiments. This helps to reduce the amount of

clusters gained by DBSCAN, while preserving relevant clusters.

3.2.5 Spectral Clustering

Spectral Clustering is a graph based algorithm based on a similarity measure between data points without assuming any knowledge of the underlying data distribution. This approach calculates a similarity matrix which is used to cluster data points according to their similarity measure with respect to the given number of clusters k.

3.2.5.1 Basic Algorithm

Algorithm 7 Spectral Clustering

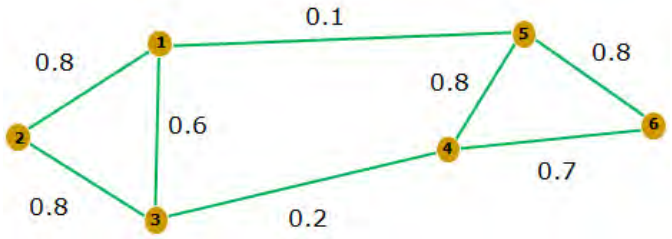
Require: number of cluster k
1: calculate similarity matrix W
2: calculate diagonal matrix D
3: $L \leftarrow D - W$
4: calculate first k eigenvectors
5: cluster data points in the k-dimensional space generated by the eigenvectors
6: **return** data points and their corresponding cluster

Spectral Clustering is based on the use of a similarity graph, the basic algorithm is shown in Algorithm 7. Data points are represented by nodes and a similarity between all data points is calculated and stored in a similarity matrix. If nodes are sufficiently similar, they are connected by an edge weighted with the similarity (Figure 3.9a). Hence nodes within a group have high weights, nodes from different have low weights. Some popular similarity graphs mentioned by [24] are the ϵ-neighborhood graph (all nodes whose distance is smaller than ϵ are connected), the k-means neighbor graph (nodes are connected with their k nearest neighbors) and the fully connected graph (all nodes are connected).

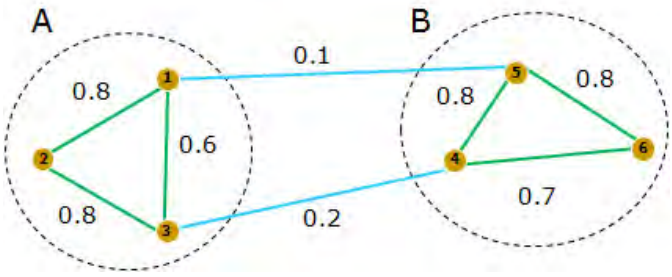
To cluster nodes according to their weighted edges, a graph cut through the edges having lower weights must be performed (Figure 3.9b). This can be done by calculating the so called graph Laplacian matrix L, defined as

$$L = D - W \tag{3.4}$$

where D is the diagonal degree matrix with the sum of adjacent weights on the diagonal and W is the quadratic weight matrix. Afterwards, the first k eigenvectors with the smallest corresponding eigenvalues are calculated from L which results in a matrix



(a) Similarity graph



(b) Graph cut

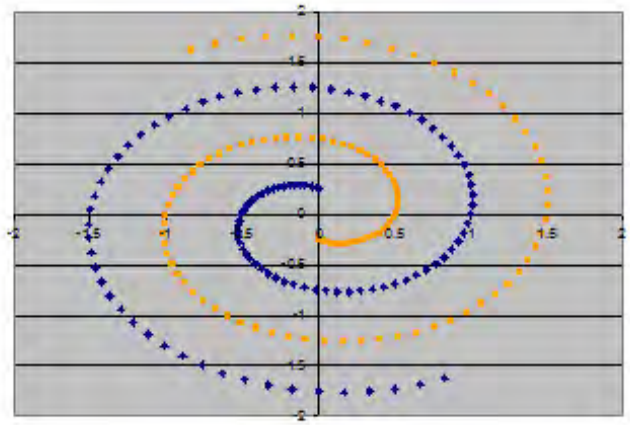
Figure 3.9: Neighborhood graph [4]

containing k n-dimensional vectors. Using row vectors instead of column vectors reduces the dimensionality of data from n-dimensional space to a k-dimensional space where clustering is much easier and the simple k-means algorithm performs very well (Figure 3.10).

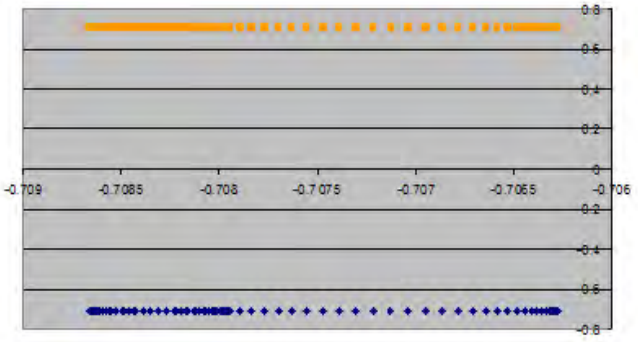
3.2.5.2 Improvements

A common algorithm widely used for spectral clustering is the Ng-Jordan-Weiss algorithm [25], which uses a scale factor σ for calculating the affinity matrix. This scale factor stays constant for all data points, hence the selection of σ to achieve desired results is challenging (Figure 3.11). [5] introduced local scaling: the scale factor σ is not constant for the whole data set, but assigning a local scale factor σ_i results in high affinities within clusters and low affinities between clusters yielding in improved clustering results.

A further improvement is the additionally added automatic detection of cluster numbers k called self-tuning spectral clustering proposed by [5], where using local scaling and analyzing the eigenvectors lead to the number of clusters. Therefore the graph Laplacian matrix L is rearranged to be as block diagonal as possible, meaning that each block is the representation of one cluster.



(a) Data points in original space



(b) Data points in new space given by k eigenvectors

Figure 3.10: Sample data set [4]

3.2.6 Growing Neural Gas

Growing Neural Gas (GNG) is a graph based approach for clustering n dimensional data, which inserts and removes nodes and edges dynamically to fit the underlying data structure. Hence no prior information about the number of nodes (clusters) needs to be specified, but other parameters which influence the number of nodes indirectly have to be adjusted. This approach is based on Competitive Hebbian Learning (CHL, [26]), which constructs a topological structure using the vector quantization of Neural Gas

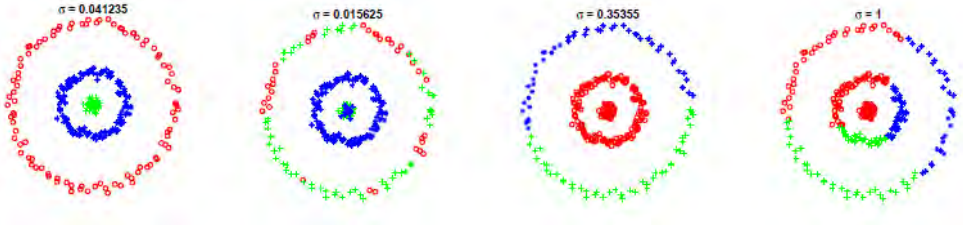


Figure 3.11: Clustering results using different values for σ [5]

(NG, [27]). GNG overcomes the problem of specifying the number of clusters a priori as when using NG and CHL.

3.2.6.1 Basic Algorithm

Algorithm 8 GNG

```
randomly place two nodes and connect them by an edge
i ← 0
for all edges do
    age ← 0
end for
repeat
    choose an arbitrary data point x̄
    find nearest node s and second nearest node t to x̄
    update local error error_s = error_s ← ‖w̄_s − x̄‖²
    move s and corresponding neighbor nodes towards x̄
    for all edges do
        age ← age + 1
    end for
    if s and t are connected then
        age(edge_st) ← 0
    else
        insert edge between s and t
    end if
    if i is a multiple of λ then
        insert new node
    end if
    multiply errors by a factor
    i ← i + 1
until maximum number of nodes or iterations is reached
return data points and their corresponding cluster
```

The basic algorithm of GNG is shown in Algorithm 8. Each cluster is represented by a node called reference vector describing the cluster center $\bar{w_s}$. Alternatively, the connected components of a graph can be used as cluster representation as well. Every node also contains information about the local error and its neighborhood. Information about neighborhood is stored as the age of edges, connecting the current node with a neighbor node. All edges exceeding a maximum age will be removed – if this is the last edge of a node, this node is removed as well. The following steps are repeated, until the maximum number of nodes or iterations are reached.

The first step sets two nodes randomly in the n dimensional space and connects them with an edge [28]. Given a data point $\bar{x}$, the nearest node s and the second nearest node t are calculated using the Euclidean distance. The error of s is updated with

$$error_s = error_s + \|\bar{w}_s - \bar{x}\|^2 \tag{3.5}$$

so it is increased by the Euclidean distance.

The second step moves the node s and its corresponding neighbor nodes connected by an edge towards $\bar{x}$. This is done by adding the difference $\bar{x} - \bar{w}_s$, respectively $\bar{x} - \bar{w}_i$ ($\bar{w}_i \in neighborhood$), multiplied by a factor between 0 and 1. This factor chooses the flexibility of moving nodes towards points. If it is too high, nodes will change their positions very often and their oscillation will be high – the smaller the factor, the more stable the graph acts due to the smaller movements, but the number of iterations increases as prototypes need longer to converge. The ages of all edges from node s are incremented, because the reference vector changes its position and its neighborhood might change as well; hence it is only valid for a certain time. If s and t are connected, the age of this edge is set to 0; otherwise a new edge between s and t is created.

Every λ iterations, a new node is inserted into the graph. In order to reduce the overall error as much as possible, the new node is placed between the node with the largest current error estimate and its neighbor with the maximal error variable. Finally, all errors are multiplied by a factor α (newly inserted node and nodes connected through newly inserted edges) and β (all other nodes) to reduce the error and to ensure that recent error estimates are weighted higher than older ones.

3.2.6.2 Improvements

GNG is very sensitive to the chosen input parameter and tends to under- or overestimate the number of clusters, due to the fixed insertion rate λ. Hence, the use of minimum description length (MDL) in combination with GNG suggested by [29] yields in better results, as it reduces the graph complexity dramatically.

Each cluster is represented by a reference vector describing the cluster center – the set of all reference vectors and associated errors (Equation 3.5) is called *code book*. This approach tries to minimize the costs of describing the code book through eliminating unnecessary reference vectors. At first, outliers are removed from the code book – outliers are reference vectors, where the costs of coding the data point directly would be lower than by describing the data point with a reference vector and its corresponding error. Afterwards, the reduction of description length is done by eliminating a reference vector, which causes the maximal reduction in description length. All data points associated with this reference vector are now associated with their second nearest reference vector

and their errors are recalculated. These two steps are processed repeatedly until no outlier and no reference vector can be removed any more.

Chapter 4

Experimental Results: Particle Advection

To obtain highest quality results for accurate source and sink detection, parameter settings of Particle Advection and clustering algorithms have to be optimized and the influence of different parameters has to be evaluated. This Chapter discusses the evaluation of different settings for the particle advection process, different settings for clustering algorithms are discussed in Chapter 5. Figure 4.1 depicts an organization chart of this Chapter. Section 4.1 shows results of different parameter settings for particle advection using image coordinates. The improvement of using world coordinates instead of image coordinates is evaluated in Section 4.2. Using an ellipse as stranded criterion is evaluated in Section 4.3, particle hopping detection in Section 4.4 and the hierarchical approach is evaluated in Section 4.5.

The aim of this Chapter is to find "optimal" particle advection settings through maximizing the percentage of valid trajectories by evaluating the influence of different particle advection settings on the quality of trajectories. The usefulness of the developed approaches to enhance the quality of particle advection introduced in Section 2.4 can also be compared by using the percentages of valid trajectories.

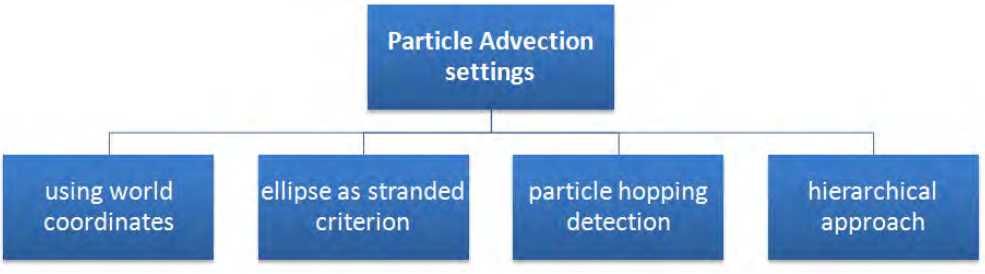


Figure 4.1: Organization chart of Chapter 4

Parameter	Value
diffusion α	5.0
diffusion q	0.5
diffusion σ	2.0
diffusion tensor	true
gauss tv ε	0.01
λ	40
median filter	true
number of iterations	10
number of warps	5
scale factor	0.7
model	TVL1
filter	none
weight	0.8
smoothing	5.0
texture rescale	true
Θ	0.1
warp direction	forward

Table 4.1: Settings for Optical Flow Detection [14]

Definition 4.1. Valid trajectories are all trajectories starting in source areas and ending in sink areas.

Definition 4.2. Invalid trajectories are all trajectories not starting in source areas and ending in sink areas.

As the number of particle advection parameters is high (there are 17 different parameters for optical flow analysis and 26 parameters for the particle advection process itself), not all parameter settings can be evaluated in reasonable time. The parameters of optical flow analysis from [14] have been set to values shown in Table 4.1, yielding in good results verified by visual inspection. If only sources and sinks are detected, the choice of optical flow parameters is not critical. The four major parameters of particle advection (insertion rate[1], stranded rate, stranded radius and backward range) have been evaluated first, to enhance trajectory quality before other effects (e.g. perspective view) were taken into account. Parameter values used for evaluation are shown in Table 4.2 and 4.3. We evaluated four different insertion and stranded rates using either image or world coordinates and five or six different backward ranges (Algorithm 2, line 25).

Evaluating only values of these four parameters results in more than 320 possible combinations of these parameter values. Advanced settings like using world coordinates instead of image coordinates or the developed particle hopping detection mechanism have been evaluated using the "optimal" particle advection settings found in this step.

[1] 50×50 particles are inserted every x frames

Insertion Rate t_{insert} [frames]	Stranded Rate t_{strand} [frames]	Stranded Radius r [pixel]	Stranded Radius r [cm]	Backward Range μ [frames]
20	10	2	25	20
50	40	5	50	60
80	70	8	75	100
110	100	11	100	140
				180

Table 4.2: Settings for Particle Advection evaluating train station video using image and world coordinates

The particle advection algorithm has been evaluated on a dense crowded train station video, on crowded PETS (Performance Evaluation of Tracking and Surveillance) benchmark data and on a crowded traffic video provided by the University of Central Florida crowd data set. To get an impression of these videos, 9 screenshots of each video have been made. Figure 4.2 and 4.3 show a Viennese train station from two different positions - this data was also used in Chapter 3 and video footage contains over 15000 frames per video showing 15 minutes of travelers moving within the train station. It took the particle advection algorithm 15 minutes to advect particles throughout the video data on a Q6600 Quad Core Processor with 2.4 GHz each, 3 GB of RAM and an Nvidia GeForce 8800 GTX, facilitating real-time particle advection on this machine. The PETS video, depicted in Figure 4.4, contains 232 frames and shows a group of people entering the camera field of view, moving from left to right and finally leaving the camera field of view. The scene in Figure 4.5 depicts a dense traffic scene containing 608 frames and obstacles thus making the detection of real sources and sinks challenging.

Figure 4.6 depicts the areas of interest of a train station and a PETS benchmark video. The areas of sources and sinks have been predefined by visual inspection of the video data (rectangular areas), facilitating the classification into valid or invalid trajectories. The main source and sink found by visual inspection due to the arrival of a train are marked cyan in figure 4.6a. Green sources and sinks are primary sources and sinks, yellow sources and sinks mark secondary sources and sinks.

Insertion Rate t_{insert} [frames]	Stranded Rate t_{strand} [frames]	Stranded Radius r [pixel]	Stranded Radius r [cm]	Backward Range μ [frames]
20	10	2	25	20
50	40	5	50	50
80	70	8	75	80
110	100	11	100	110
				140
				170

Table 4.3: Settings for Particle Advection evaluating PETS benchmark data using image and world coordinates

(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

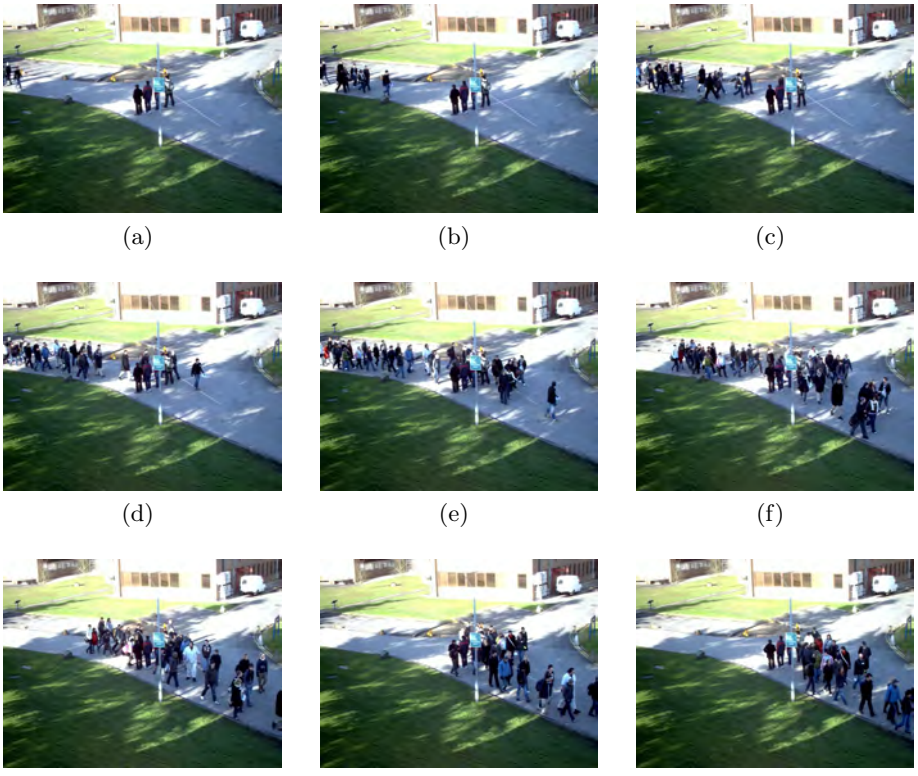
Figure 4.2: Screenshots of train station video - position 1

(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

Figure 4.3: Screenshots of train station video - position 2

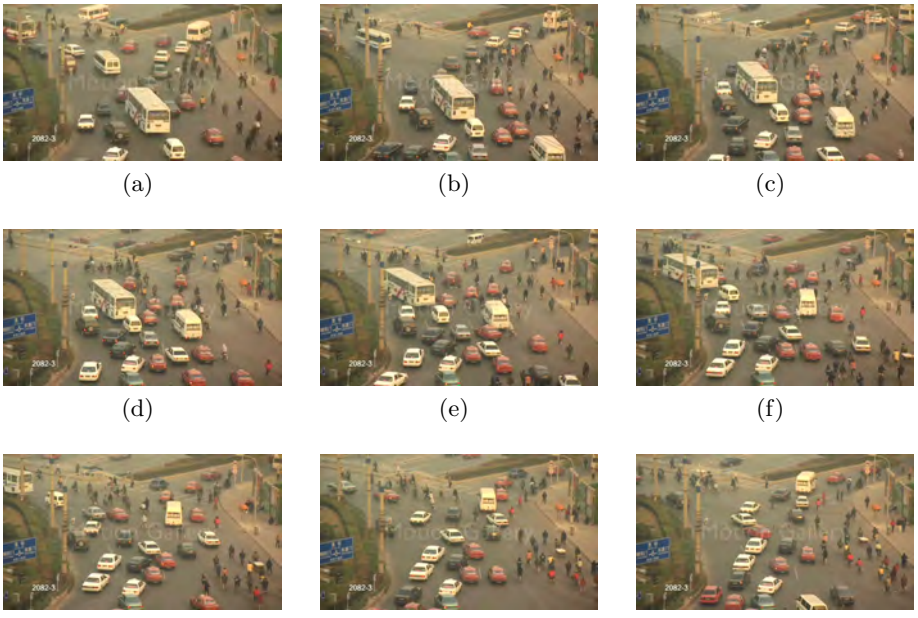


(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

Figure 4.4: Screenshots of PETS video



(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

Figure 4.5: Screenshots of traffic video

(a) Train station (b) PETS benchmark data

Figure 4.6: Sources and sinks

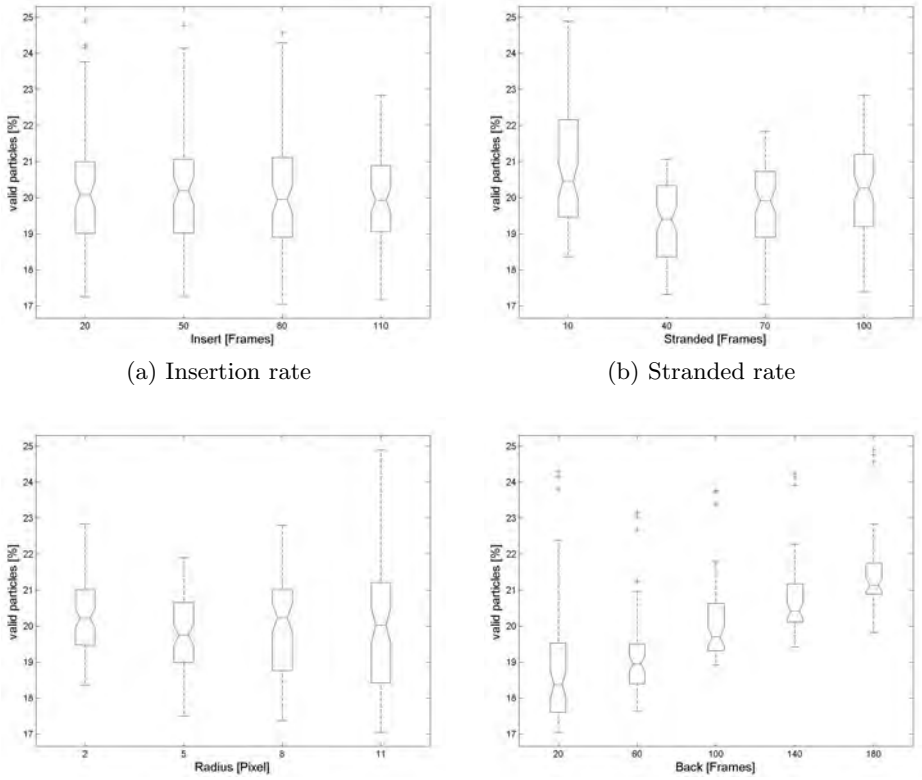
4.1 Particle Advection Settings using Image Coordinates

Using image coordinates as dimension unit may result in problems caused by perspective distortion, depending on the camera position introduced in Section 2.3.2. This Section provides data obtained by using the basic particle advection algorithm without any enhancements.

4.1.1 Parameter Settings for Train Station

Using boxplots, the mean value, quartiles and outliers of 320 runs using different parameter settings are calculated, shown in Figure 4.7. These plots show that only the backward range does have a clear major effect on the quality of trajectories. Since the notches in the boxplot do not overlap, one can say with 95% confidence that the backward range does influence the quality significantly. There is also a influence of the stranded rate, but this dependency is not significant.

The second step of evaluation was a qualitative evaluation comparing start and end points of different particle advection settings using a kernel density estimator introduced in Chapter 3. Qualitative evaluation has shown that a large backward range and a high insert rate result in more intensified sources and sinks. Setting a large backward range means that each particle which is inserted into the system, is advected backwards for the specified number of frames before it is advected forward. The longer the particle is advected backwards, the higher the probability of finding the "real" source of this particle. Thus, particles should be advected backward as long as possible (e.g. 180 frames) shown Figure 4.8. The number of frames, a particle could be advected backward, is restricted by main memory because every optical flow field has to be hold in main memory, as it is needed for backward advection. Advecting particles back for a long



(a) Insertion rate

(b) Stranded rate

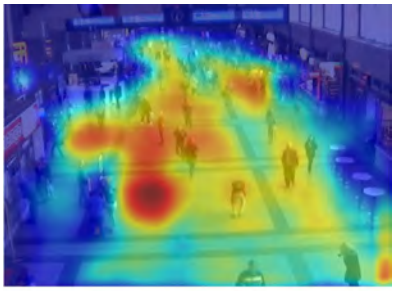
(c) Stranded radius

(d) Backward range

Figure 4.7: Quality of trajectories depending on particle advection settings (II)

time using a relaxed stranded criterion results in a higher number of particles which can cause a lack of memory either. Backward advection does not have any influence on the obtained sinks, as only the start point is changed during backward advection, which has been verified in Figure 4.9. Figure 4.9 depicts the results of using different backward ranges, which do not have any influence on the obtained sinks.

Using a high insert rate (e.g. every 20 frames new particles are inserted), meaning that as many particles as possible (depending on the size of main memory) are in the system results in more intensified sources and sinks, illustrated in Figure 4.10 and Figure 4.11. This may be due to the fact of the higher sample that can be evaluated. Inserting particles every 20 frames results in over 1.2 million particles when analyzing the train station video, which contains more than 15.000 frames. As the memory consumption directly depends on the number of particles, the number of particles is restricted by available main memory.



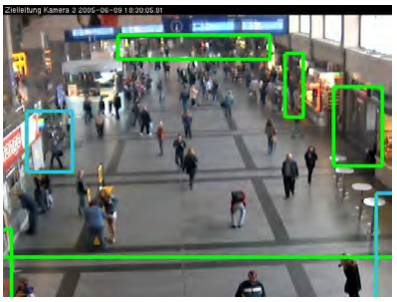
(a) Backward range = 20

(b) Backward range = 60

(c) Backward range = 100

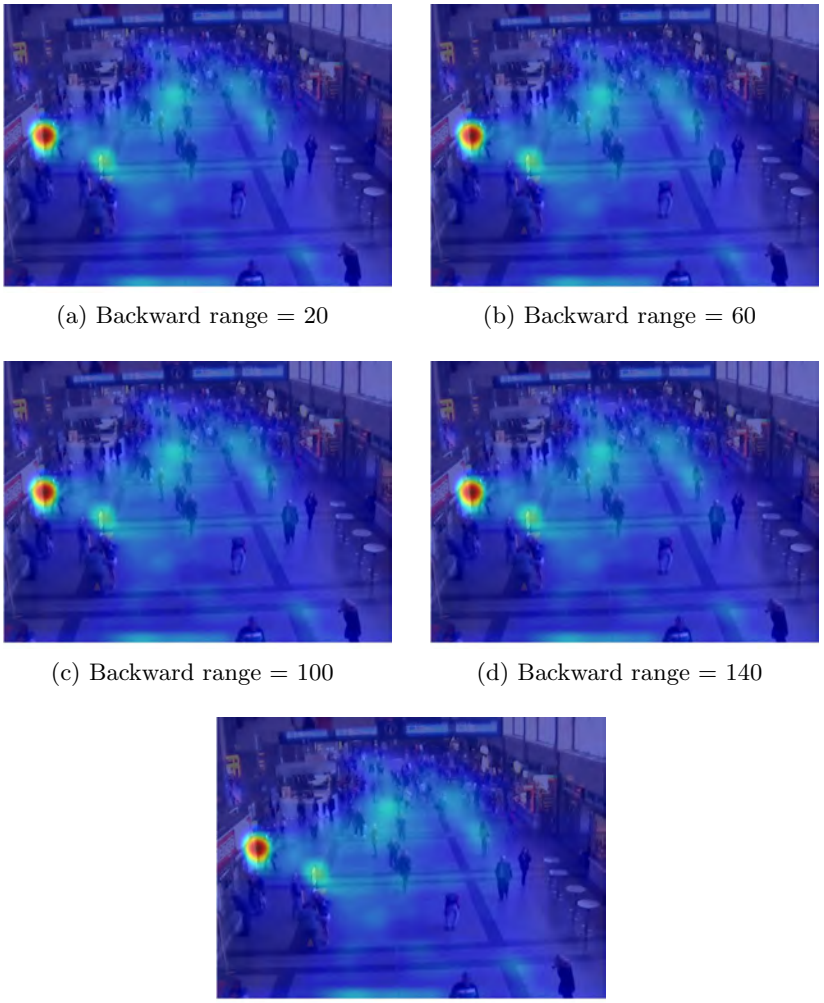
(d) Backward range = 140

(e) Backward range = 180



(f) Ground truth - main source was identified by visual inspection to be in the right bottom corner (cyan)

Figure 4.8: Influence of backward range on trajectory start points (insert rate = 20, stranded rate = 10, stranded radius = 2)



(a) Backward range = 20

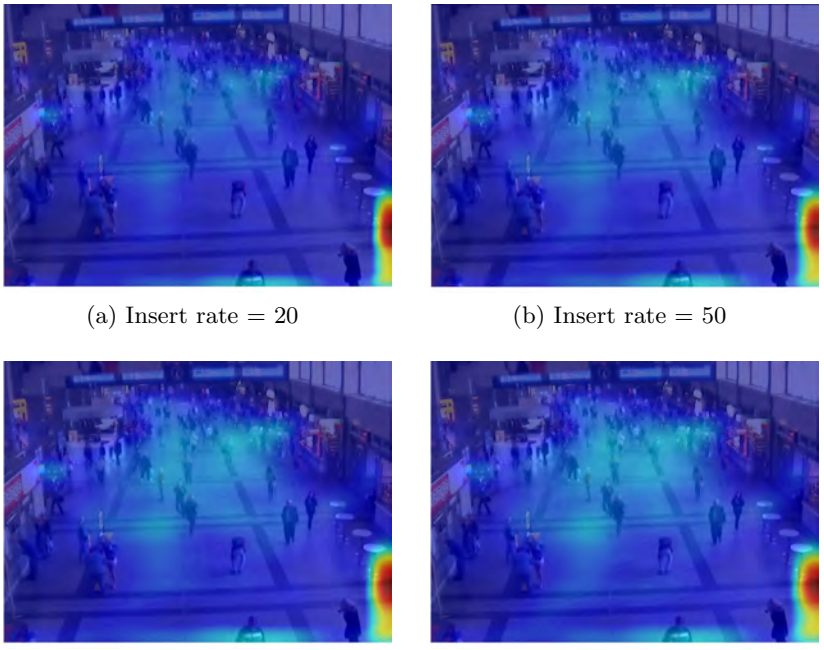
(b) Backward range = 60

(c) Backward range = 100

(d) Backward range = 140

(e) Backward range = 180

Figure 4.9: Influence of backward range on trajectory end points (insert rate = 20, stranded rate = 10, stranded radius = 2)



(a) Insert rate = 20

(b) Insert rate = 50

(c) Insert rate = 80

(d) Insert rate = 110

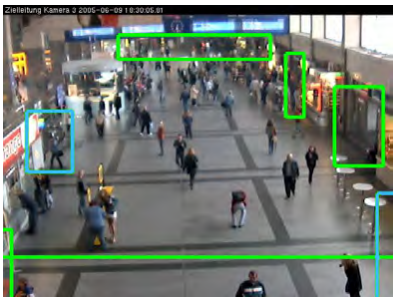
Figure 4.10: Influence of insertion rate on trajectory start points (stranded rate = 10, stranded radius = 2, backward range = 180)

(a) Insert rate = 20

(b) Insert rate = 50

(c) Insert rate = 80

(d) Insert rate = 110



(e) Ground truth - main sink was identified by visual inspection to be an escalator on the left side (cyan)

Figure 4.11: Influence of insert rate on trajectory end points (stranded rate = 10, stranded radius = 2, backward range = 180)

A very relaxed stranded criterion also leads to a high number of particles thus intensifying sources and sinks – shown in Figure 4.12 and Figure 4.13. To relax the stranded criterion, the stranded radius is high and the stranded rate should be low, meaning that a particle strands after a large number of frames (e.g. particle strands after 100 frames, if it only moves within a radius of eleven pixels). Not stranding immediately causes the particles to hop from one person to another one. When the two persons head into the same direction, this "particle hopping" is not critical as we are only interested in general motion flows.

These results lead to the following guidelines for the choice of particle advection parameters:

- large backward range
- high number of particles, implicating a
- high insertion rate and a
- relaxed stranded criterion

Using these guidelines leads to intensified sources and sinks, but requiring more main memory.

4.1.2 Parameter Settings for PETS Benchmark Data

Figure 4.14 depicts the quantitative analysis of a PETS benchmark video, showing the percentage of primary trajectories – these are trajectories which start at a primary source and end in a primary sink. As the results obtained by analysis of the train station video are general guidelines, the results of the PETS benchmark data should confirm these guidelines.

However, Figure 4.14 does not show any clear dependencies of any parameter on the quality of particle trajectories. This is caused by the short length of the PETS video of only 232 frames. Results obtained by longer video analysis are more trustful than results obtained by analysis of a short video sequence. The general purpose of this work is finding sources and sinks – therefore the analysis is more useful on a longer sequence than 232 frames.

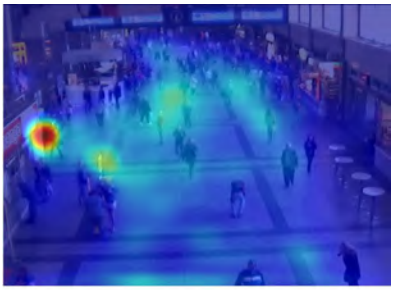
(a) Stranded rate = 10, radius = 2

(b) Stranded rate = 40, radius = 5

(c) Stranded rate = 70, radius = 8

(d) Stranded rate = 100, radius = 11

Figure 4.12: Influence of stranded rate and radius on trajectory start points (insert rate = 50, backward range = 180)



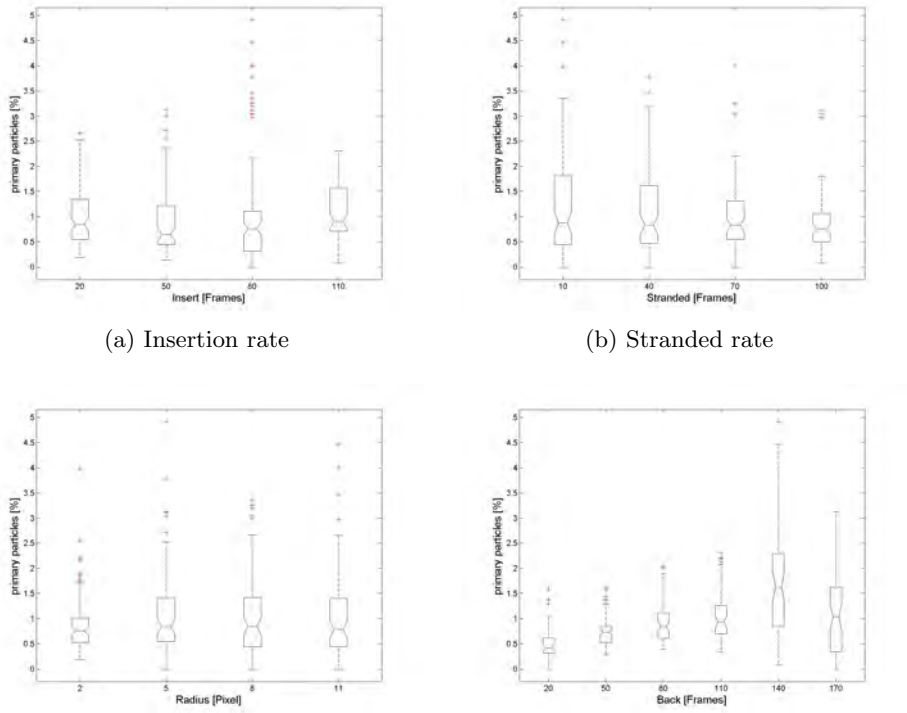
(a) Stranded rate = 10, radius = 2

(b) Stranded rate = 40, radius = 5

(c) Stranded rate = 70, radius = 8

(d) Stranded rate = 100, radius = 11

Figure 4.13: Influence of stranded rate and radius on trajectory end points (insert rate = 50, backward range = 180)



(a) Insertion rate

(b) Stranded rate

(c) Stranded radius

(d) Backward range

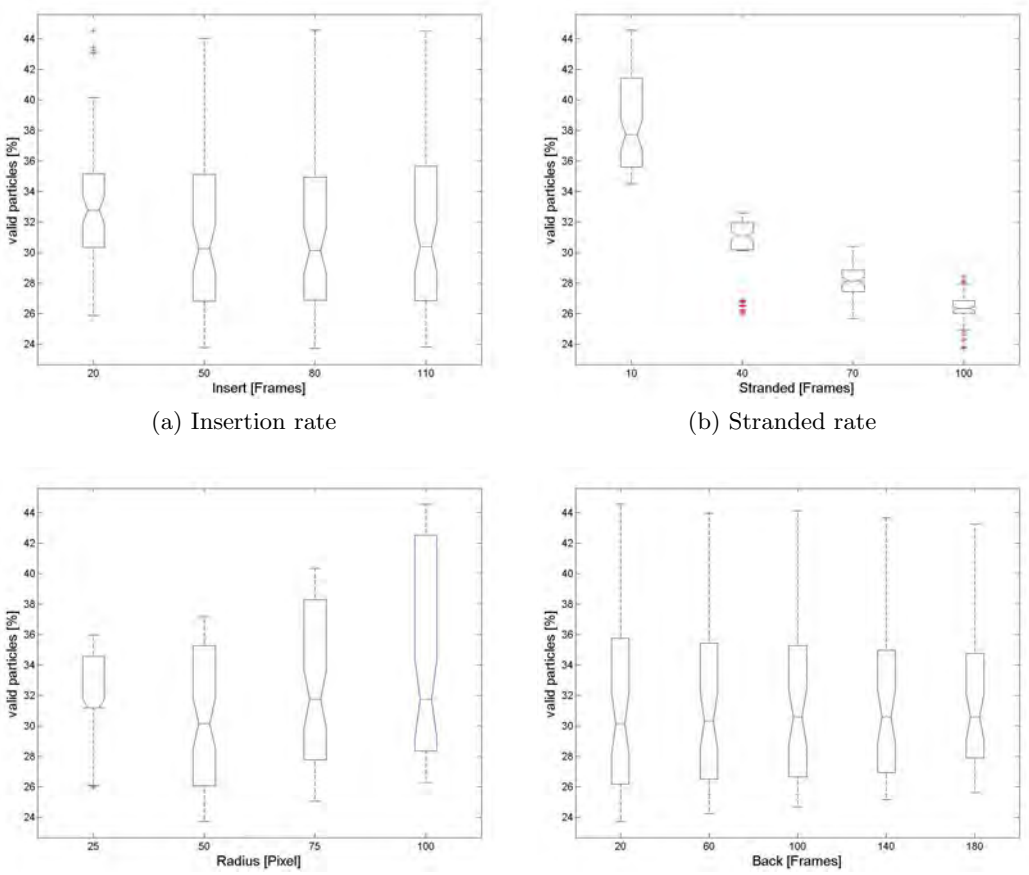
Figure 4.14: Quality of trajectories depending on particle advection settings

4.2 Particle Advection Settings using World Coordinates

To decrease the influence of perspective view and the camera position, particle advection can also employ world coordinates introduced in Section 2.3.2. This has the advantage of using a consistent stranded criterion not depending on the actual pixel position or resolution. Thus using world coordinates should yield in a higher percentage of valid trajectories.

4.2.1 Parameter Settings for Train Station

Using world coordinates results in an average percentage of 31,61% of valid trajectories, whereas trajectories obtained by using image coordinates result in an average percentage of 20,06% of valid trajectories. This comparison demonstrates that the use of perspective view (and its distortions) has a big influence on the quality – therefore the use of world coordinates is recommended (if camera calibration data exists). Figure 4.15 shows that the quality significantly depends on the stranded rate and that the size of backward range does not influence the quality any more.

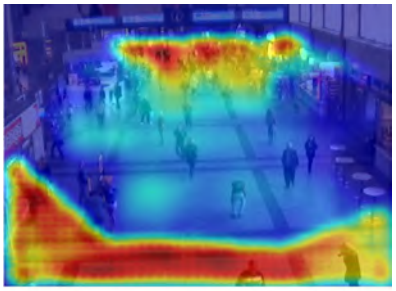


(a) Insertion rate (b) Stranded rate

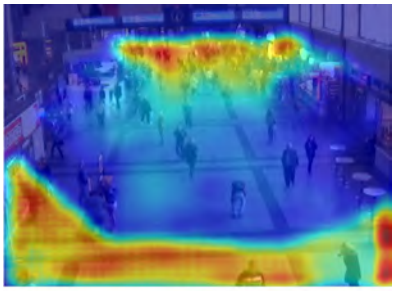
(c) Stranded radius (d) Backward range

Figure 4.15: Quality of trajectories depending on particle advection settings using world coordinates

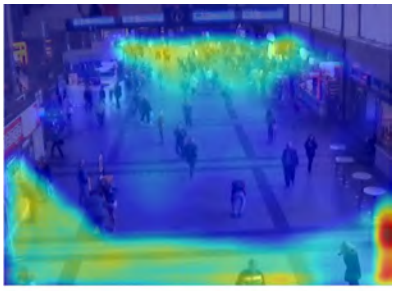
Analysis of trajectory start and end points confutes these results, because using a long backward advection intensifies sources thus influencing the quality, shown in Figure 4.16. In general, the results are very similar to those obtained by particle advection with image coordinates, thus confirming the general guidelines introduced in Section 4.1.1. Figure 4.17 and Figure 4.18 show that a higher insertion rate (particles are inserted every 20 frames) results in more intensified sources and sinks than a lower insertion rate (particles are inserted every 110 frames). A relaxed stranded criterion also results in more intensified sources and sinks as this results in more particles, since they are not removed immediately. Figure 4.19 and 4.20 depict the influence of the stranded criterion on the obtained sources and sinks: using a strict stranded criterion (particles are removed after ten frames if they move within a radius of 25 cm) does not intensify the main source and sink as it does when using a relaxed stranded criterion (particles are removed after 100 frames if they move within a radius of 100 cm).



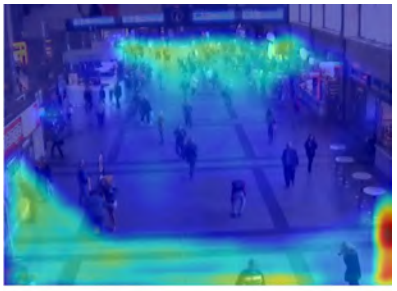
(a) Backward range = 20



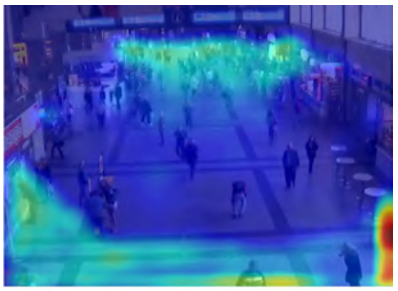
(b) Backward range = 60



(c) Backward range = 100

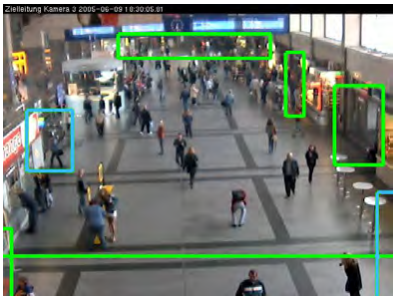


(d) Backward range = 140

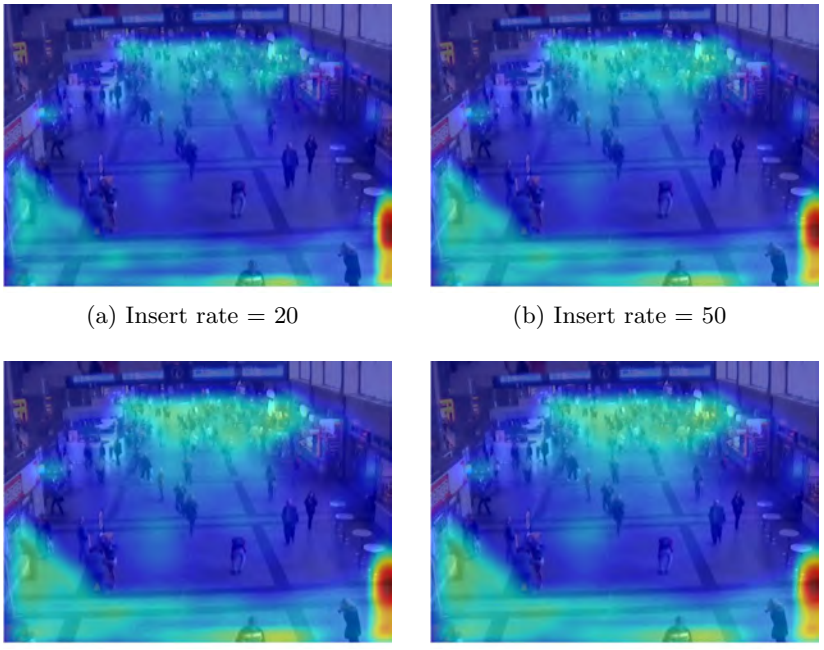


(e) Backward range = 180

Figure 4.16: Influence of backward range on trajectory start points using world coordinates (insert rate = 20, stranded rate = 10, stranded radius = 25)



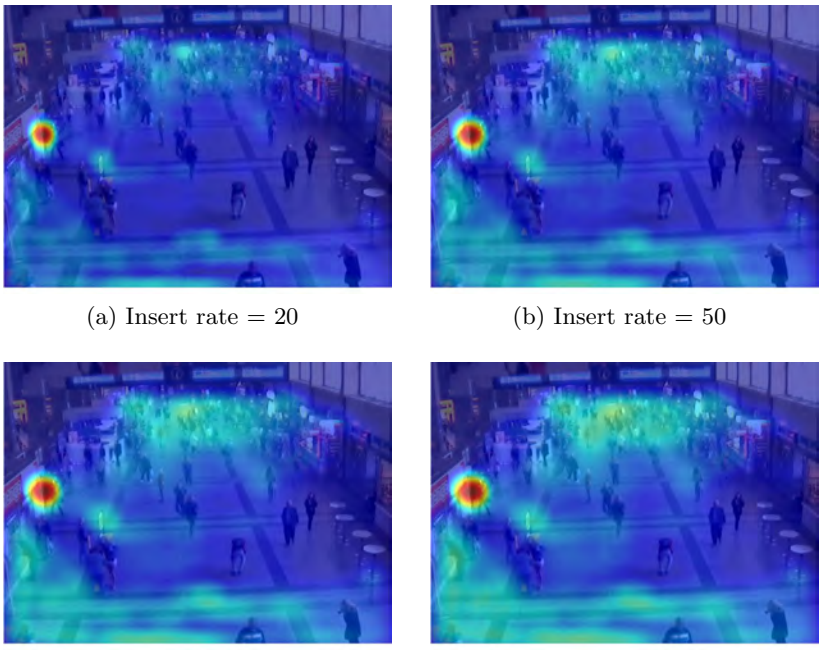
(a) Ground truth - main source was identified by visual inspection to be in the right bottom corner (cyan)



(a) Insert rate = 20 (b) Insert rate = 50

(c) Insert rate = 80 (d) Insert rate = 110

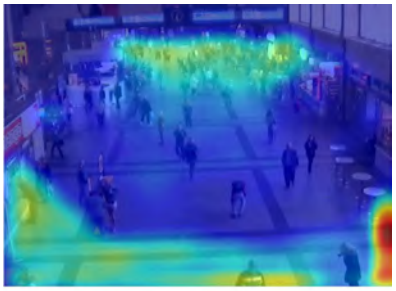
Figure 4.17: Influence of insert rate on trajectory start points using world coordinates (stranded rate = 40, stranded radius = 50, backward range = 180)



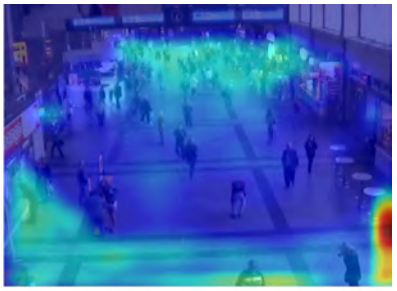
(a) Insert rate = 20 (b) Insert rate = 50

(c) Insert rate = 80 (d) Insert rate = 110

Figure 4.18: Influence of insert rate on trajectory end points using world coordinates (stranded rate = 40, stranded radius = 50, backward range = 180)



(a) Stranded rate = 10, radius = 25

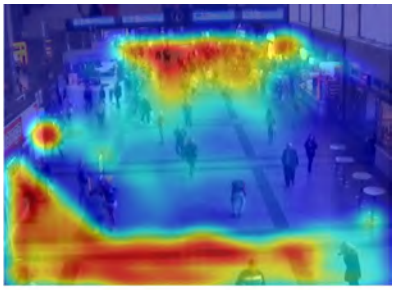


(b) Stranded rate = 40, radius = 50

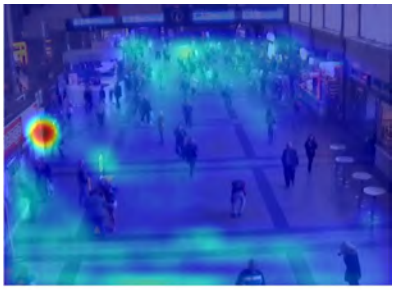
(c) Stranded rate = 70, radius = 75

(d) Stranded rate = 100, radius = 100

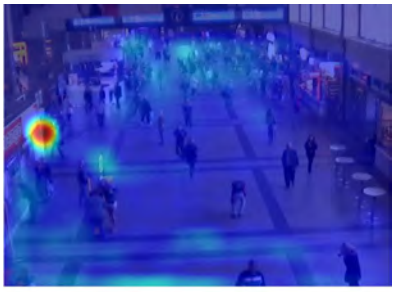
Figure 4.19: Influence of stranded rate and radius on trajectory start points using world coordinates (insert rate = 50, backward range = 180)



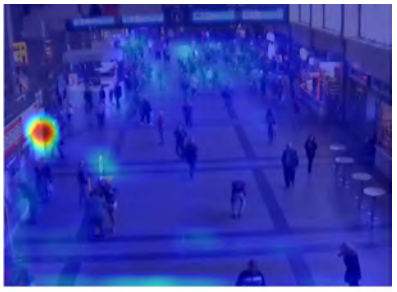
(a) Stranded rate = 10, radius = 25



(b) Stranded rate = 40, radius = 50

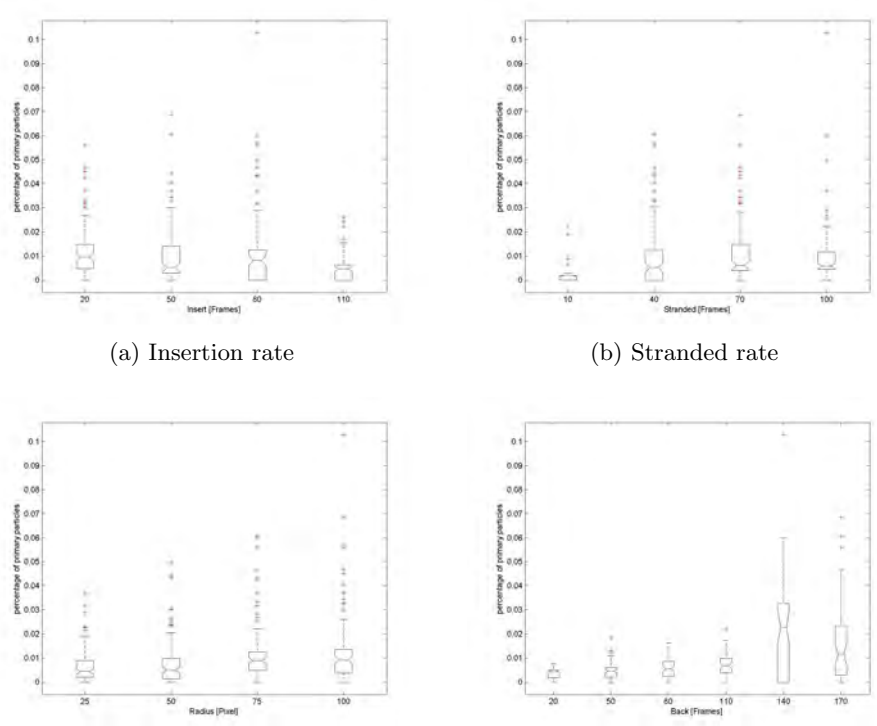


(c) Stranded rate = 70, radius = 75



(d) Stranded rate = 100, radius = 100

Figure 4.20: Influence of stranded rate and radius on trajectory end points using world coordinates (insert rate = 50, backward range = 180)



(a) Insertion rate

(b) Stranded rate

(c) Stranded radius

(d) Backward range

Figure 4.21: Quality of trajectories depending on particle advection settings

4.2.2 Parameter Settings for PETS Benchmark Data

The quantitative evaluation of PETS benchmark data is shown in Figure 4.21. As the benchmark data contain only 232 frames and thus being too short for a reliable analysis, no assumptions about dependencies between any parameter and the quality of particle trajectories can be made.

4.3 Ellipse as Stranded Criterion

To evaluate the use of ellipses as stranded criterion, different combinations of axis lengths were applied – Table 4.4 and 4.5 show the possible combinations. Based on evaluation done in Section 4.2, the insert rate and the backward range should be high (inserting particles every 50 frames, advecting them 180 frames back in time), while using world coordinates and a relaxed stranded criterion (particles strand after 100 frames).

Figure 4.22 depicts the percentage of valid trajectories obtained while using an ellipse as stranded criterion. The highest percentage can be found when using a long y-axis (400 cm); using a long x-axis (400cm) results in a lower percentage of valid trajectories. Hence, the uncertainness in y direction is bigger than in x direction, as a more relaxed

	x-axis [cm]			
	25	50	75	100
y-axis [cm]	50	100	150	200
	75	150	225	300
	100	200	300	400

Table 4.4: Axis Lengths for Elliptical Stranded Criterion (I)

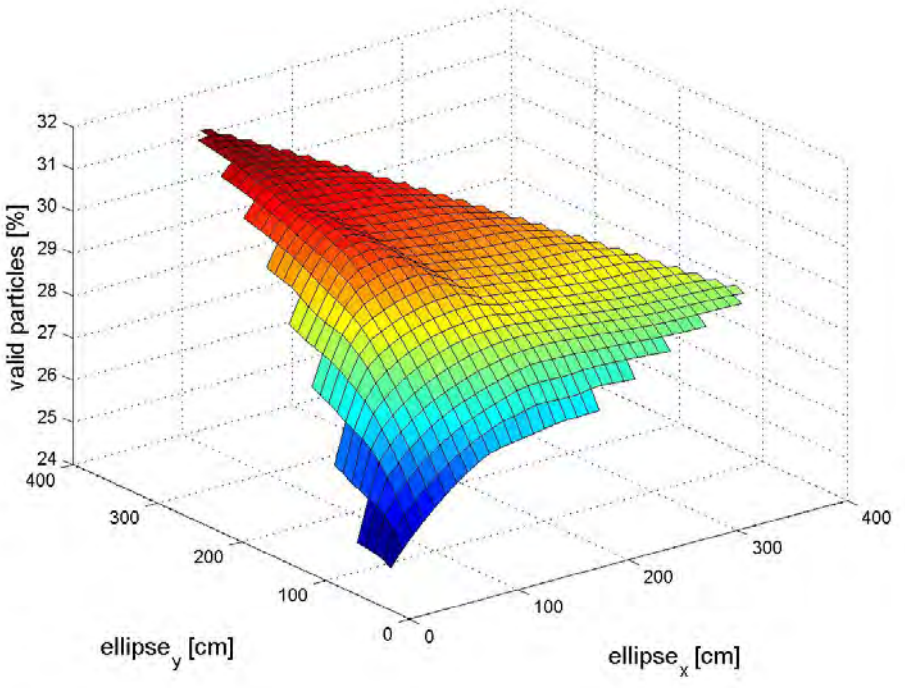
stranded criterion results in more valid trajectories. This confirms the assumption that the uncertainty in y direction is bigger as a change of one pixel in y direction results in a higher change in world coordinates than a change of one pixel in x direction.

4.4 Particle Hopping Detection

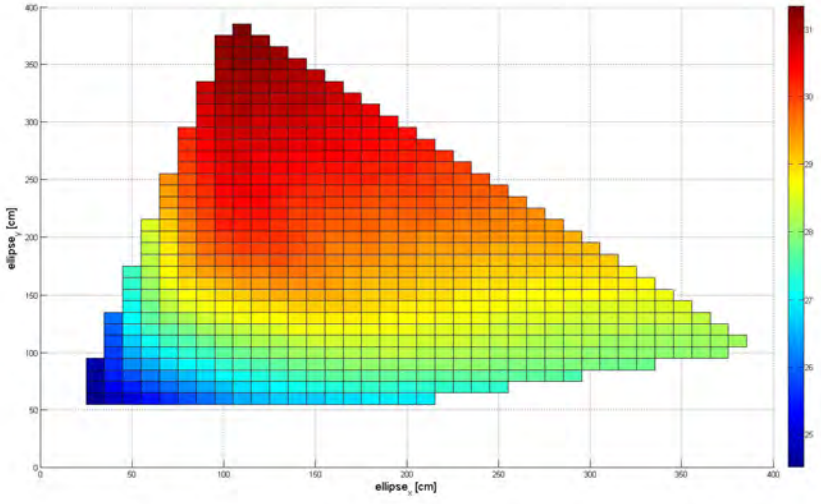
Both particle hopping detection mechanisms, restricting angle λ and acceleration, were evaluated on the train station video. As it is a dense crowded scene with many flow directions, many particle hops occurred (verified by visual inspection). By using a particle hopping detection mechanism, a higher quality of trajectories is expected meaning that the overall percentage of valid particles is higher than without using a particle hopping detection mechanism and hence sources and sinks are more distinct. The results were compared to results of the same scene without eliminating hopped particle trajectories depicted in Figure 4.23. Figure 4.23a shows the percentage of valid particles using different angles (30°, 60°, 90°, 120° and 150°) as threshold for the particle hopping detection. The blue line depicts the quality of trajectories, if no particle hopping detection mechanism is used resulting in a higher quality. The bigger the values for λ_t, the higher the quality as the criterion is more and more relaxed, depicted as red line in Figure 4.23a. Also the results using the acceleration detection mechanism in Figure 4.23b show that the quality of particles is much lower when a particle detection mechanism is used.

	y-axis [cm]			
	25	50	75	100
x-axis [cm]	50	100	150	200
	75	150	225	300
	100	200	300	400

Table 4.5: Axis Lengths for Elliptical Stranded Criterion (II)

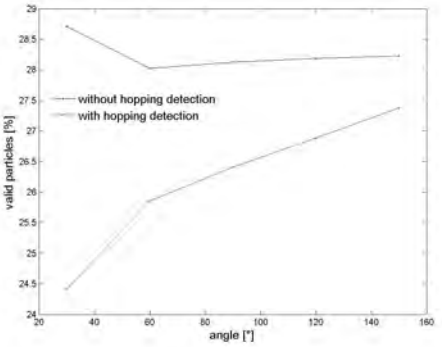


(a) 3D plot

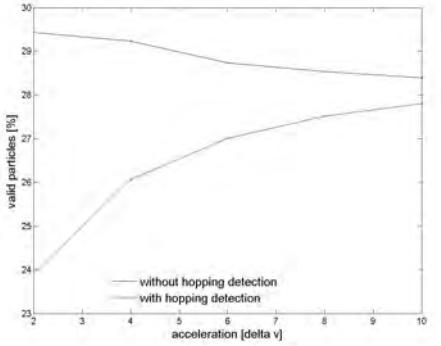


(b) 2D plot (percentage of valid trajectories coded by color)

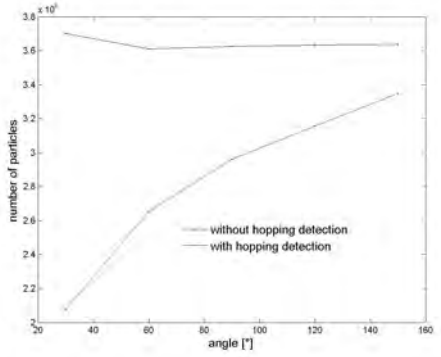
Figure 4.22: Percentage of valid particle trajectories obtained by using an ellipse as stranded criterion



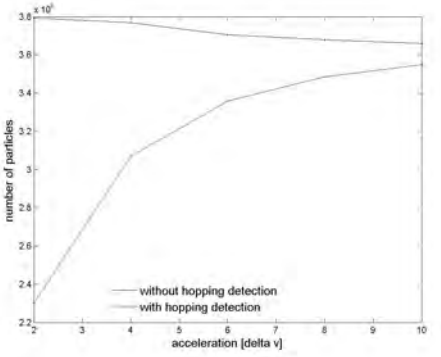
(a) Percentage of valid particles using different values for λ_t



(b) Percentage of valid particles using different values for acceleration



(c) Number of particles using different values for λ_t

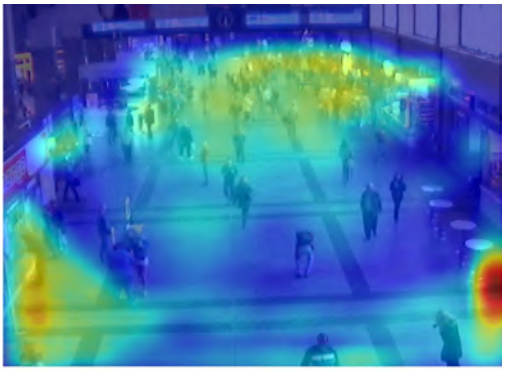


(d) Number of valid particles using different values for acceleration

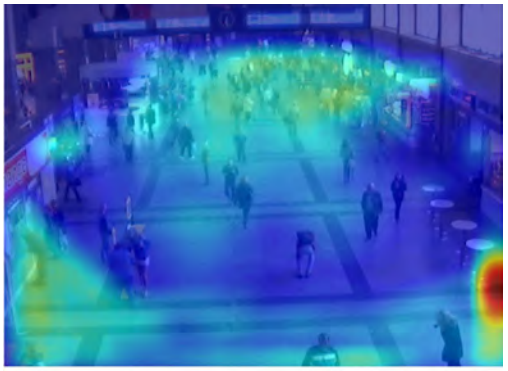
Figure 4.23: Comparison of trajectory quality and number of particles

These results are astonishing, as we would have expected higher quality trajectories. Probably this is due to the fact that using a particle hopping detection mechanism reduces the number of particles dramatically, shown in Figure 4.23c and 4.23d. Figure 4.24 shows the influence of λ_t on start points, Figure 4.25 on end points. Both Figures show that sources and sinks getting more distinctive, if a more relaxed detection is used thus confirming the results shown in Figure 4.23. Also restricting acceleration results in higher quality, if the criterion is more relaxed, depicted in Figure 4.26 and 4.27.

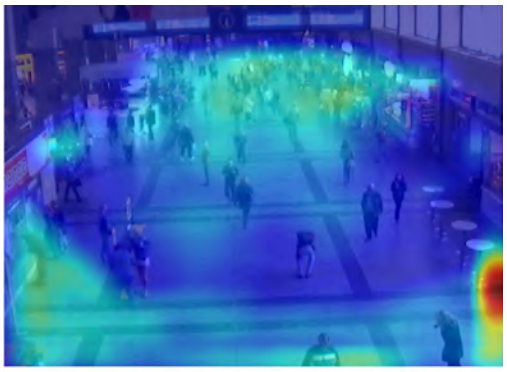
If one is interested in exact trajectory data, the use of a particle hopping detection mechanism is reasonable as trajectory data is not adulterated. The distinctiveness of sources and sinks depends on the number of particles – the higher the number, the more distinct the sources and sinks are. As a particle hopping detection mechanism reduces the number of particles respectively trajectories, sources and sinks are not so distinctive any more.



(a) $\lambda_t = 30°$



(b) $\lambda_t = 60°$

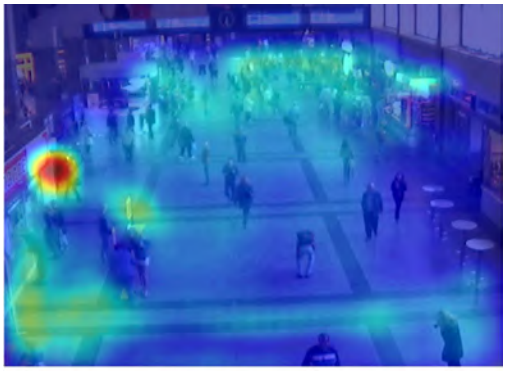


(c) $\lambda_t = 90°$

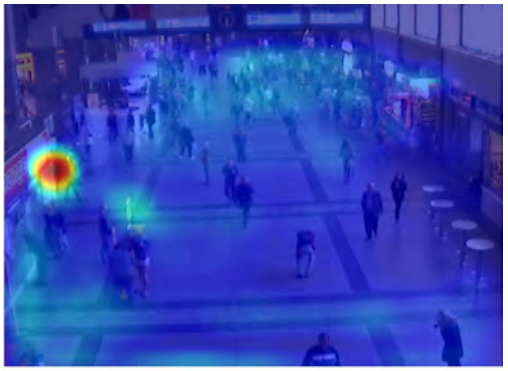
(d) $\lambda_t = 120°$

(e) $\lambda_t = 150°$

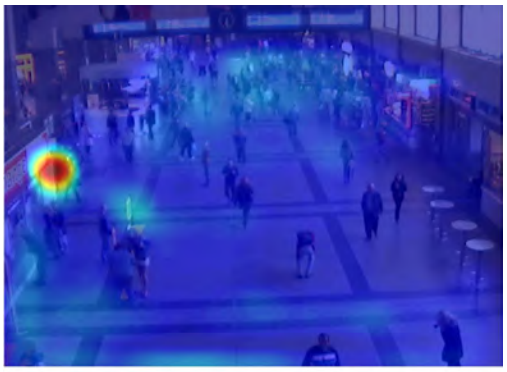
Figure 4.24: Start points using different values for λ_t



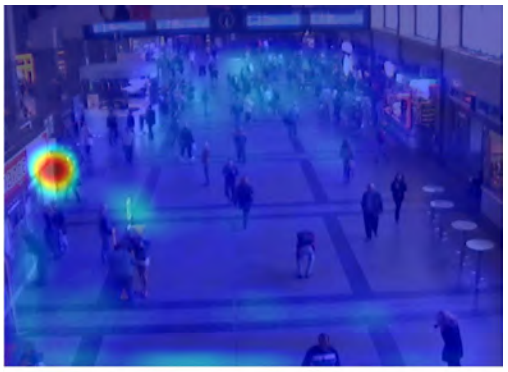
(a) $\lambda_t = 30°$



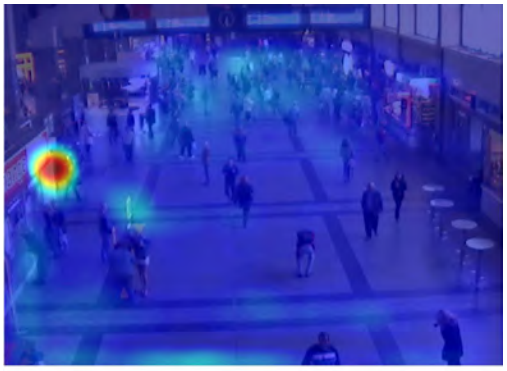
(b) $\lambda_t = 60°$



(c) $\lambda_t = 90°$

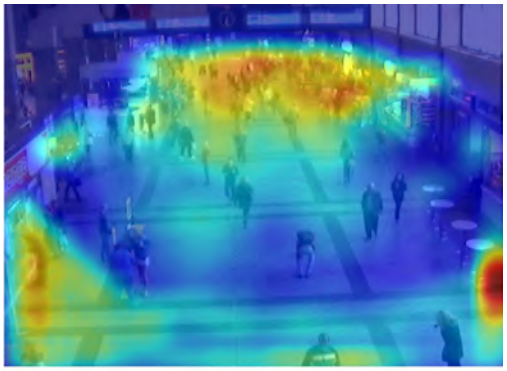


(d) $\lambda_t = 120°$

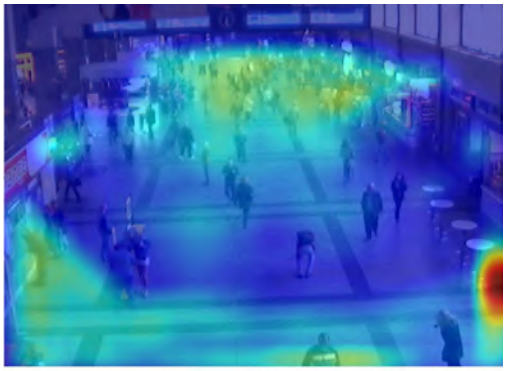


(e) $\lambda_t = 150°$

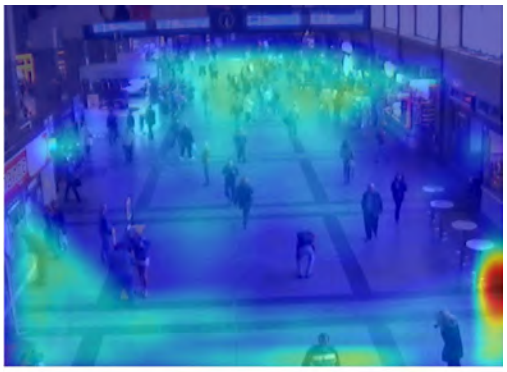
Figure 4.25: End points using different values for λ_t



(a) acceleration=2



(b) acceleration=4

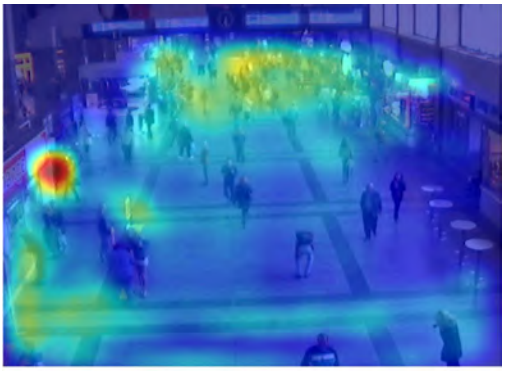


(c) acceleration=6

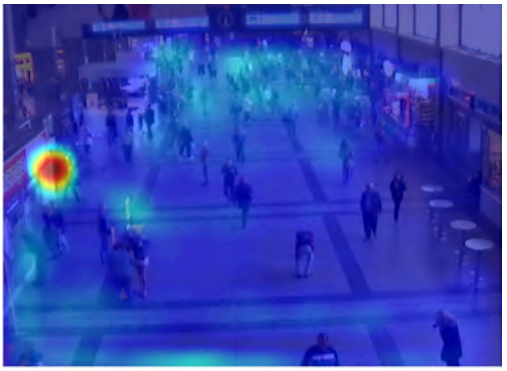
(d) acceleration=8

(e) acceleration=10

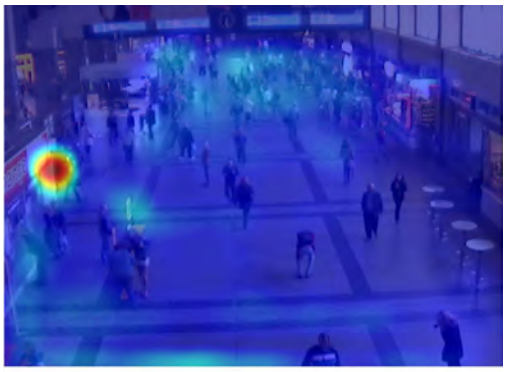
Figure 4.26: Start points using different values for acceleration



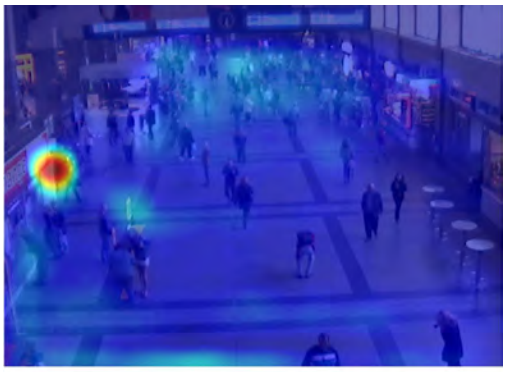
(a) acceleration=2



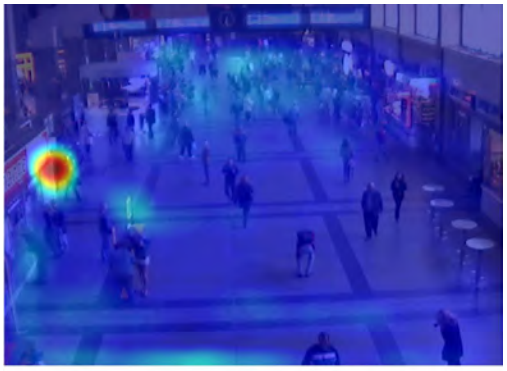
(b) acceleration=4



(c) acceleration=6



(d) acceleration=8



(e) acceleration=10

Figure 4.27: End points using different values for acceleration

4.5 Hierarchical Approach

This approach was evaluated on video data from the UCF Crowd Data Set[2]. This video was used because the train station video does not contain any static occlusions and the PETS video was to short to gain feasible results. The video contains a high density traffic scene and lasts for 608 frames.

Particles were inserted every 30 frames and were removed if they only moved within a radius of ten pixels over a period of 100 frames. Newly inserted particles were moved backward for 50 frames. Figure 4.28a shows a screenshot of the video sequence used for evaluation and the corresponding areas of interest. There is one main source at the bottom and 2 sinks in the upper left and right corner. The lamppost acts as a secondary sink as vehicles are occluded and particles tend to strand due to the lack of motion. Not using a hierarchical approach results in 12.53% of particles being valid and only 6.16% of particles are stranding in primary sinks, which is depicted in Figure 4.28b. Figure 4.28c shows trajectory end points, if the hierarchical approach is used, resulting in 13.77% of particles being valid and 7.21% of particles strand in primary sinks. If particle advection is not processed on full but on reduced resolution size, the percentage of valid particles raises (16.06% of particles are valid) and also the percentage of particles stranding in primary sinks is higher (9.66%). As the quantitative results are supported by qualitative results shown in Figure 4.28, primary sinks gain in importance while secondary sinks (lampposts) become less important. Thus one can say, the usage of hierarchical particle advection is recommandable if the exact trajectories are needed. For detecting sources and sinks of a video sequence, the use of a reduced video resolution may be sufficient, as the quality of trajectories improves with the aspect of loosing accuracy.

[2]This data set is provided by the University of Central Florida and can be found on http://www.cs.ucf.edu/∼sali/Projects/CrowdSegmentation/index.html (last accessed on February 15, 2010.)

(a) Areas of interest in UCF Crowd Dataset video

(b) End points obtained by particle advection not using the hierarchical approach

(c) End points obtained by particle advection using the hierarchical approach

(d) End points obtained by particle advection on reduced resolution

Figure 4.28: Comparing end points using different settings on hierarchical particle advection

Chapter 5

Experimental Results: Clustering Sources and Sinks

This Chapter compares results obtained by different clustering algorithms regarding cluster results, stability and the dependency on input parameters. All data points (coordinates) were normalized to fit the interval from 0 to 1 and clustering algorithms were evaluated on the same subset of start and end points. Start and end points were obtained by particle advection on the train station video, using particle advection parameters shown in Table 5.1, resulting in a high quality of trajectories (28.34% of trajectories start in sources and end in sinks).

The flowchart for clustering sources and sinks is shown in Figure 5.1. At first, data reduction is applied to all start and end points. Afterwards, different clustering algorithms are used to obtain sources and sinks. Finally, a cluster reduction process using a threshold T introduced in Equation 3.3 or Minimum Description Length introduced in Section 3.2.6.2 is used to detect the main sources and sinks.

To verify results, parameters of clustering algorithms have been verified using a second video sequence from train station, introduced in Chapter 4, Figure 4.2. Figure 5.2 depicts one main source at the right side and two main sinks – one at the left side (door

Parameter	Value
Insert Rate	20 frames
Stranded Rate	100 frames
Stranded Radius	100 cm
Backward Range	180 frames

Table 5.1: Settings for Particle Advection

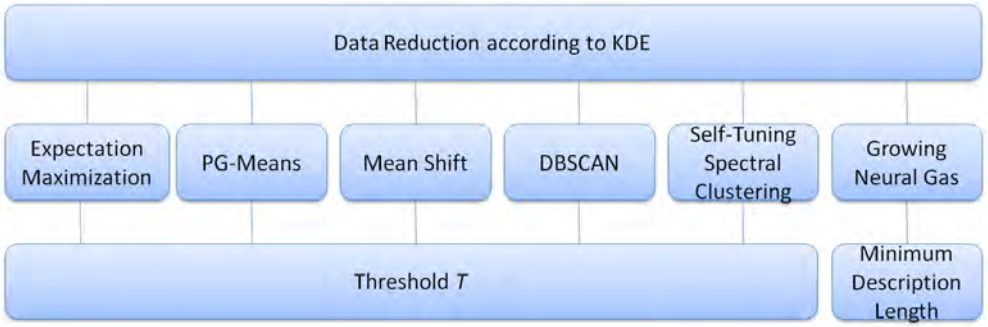


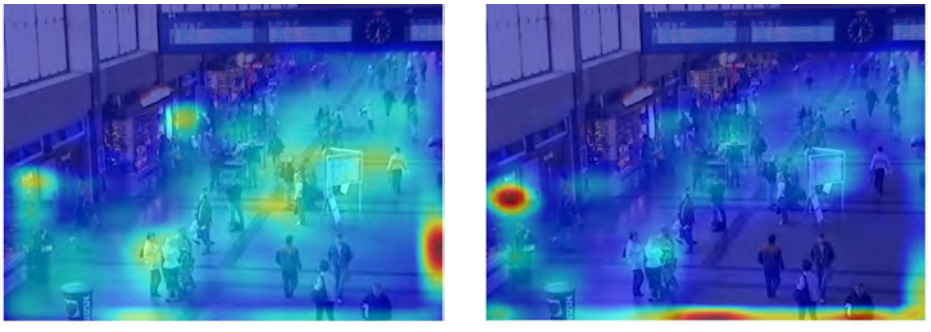
Figure 5.1: Flowchart for clustering sources and sinks

to the platform) and a larger one at the bottom, where people leave the camera field of view.

This Chapter is structured as follows: Section 5.1 describes the process of data reduction before clustering. The expectation maximization algorithm is evaluated in Section 5.2, PG-means is evaluated in Section 5.3 and the evaluation of mean shift algorithm is shown in Section 5.4. Section 5.5 depicts the experimental results of the DBSCAN algorithm, Section 5.6 the results of self-tuning spectral clustering and the evaluation of growing neural gas is shown in Section 5.7.

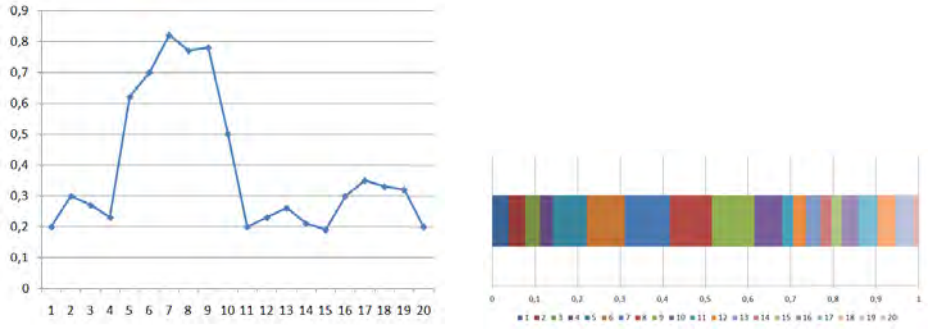
5.1 Data Reduction

Particle advection usually results in over 1 million particles, thus making clustering very time consuming, if not impossible. For example: spectral clustering uses a similarity matrix to store all similarities between all data points. Using 10^6 particles results in 10^6 trajectories, hence 10^6 start or end points have to be clustered. This results in



(a) Density of start points (b) Density of end points

Figure 5.2: Density plots of start and end points obtained by particle advection on the second train station video



(a) 20 sample pixel densities

(b) Pixel densities normalized to a range from 0 to 1

Figure 5.3: Correlation between density and probability of randomly chosen points

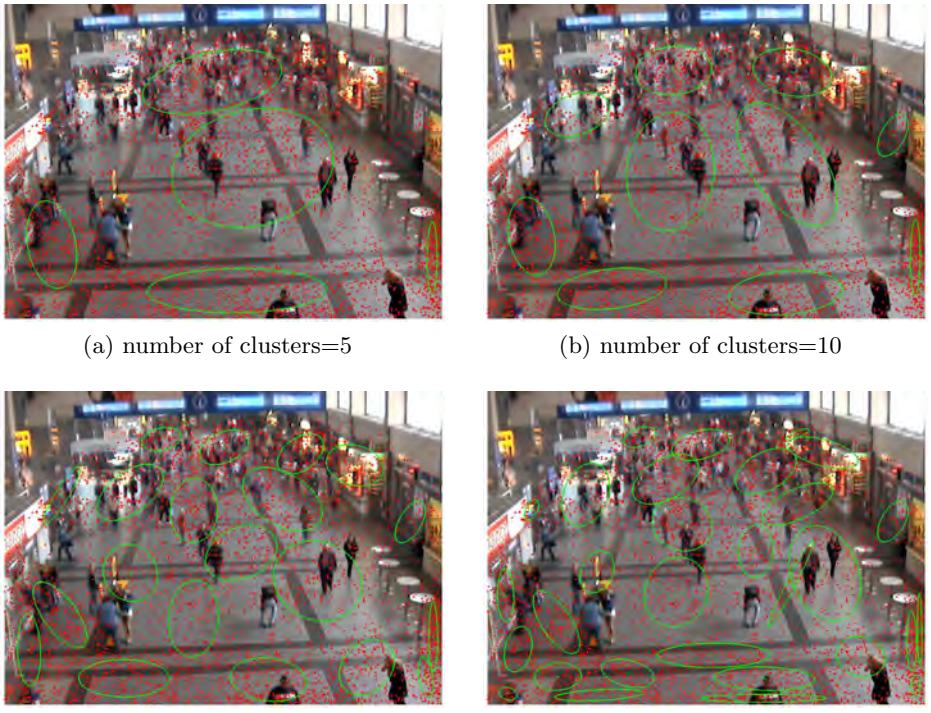
a $10^6 \times 10^6$ similarity matrix storing 10^{12} values. Using the data type double, which requires 8 bytes, the similarity matrix uses $8 \cdot 10^{12}$ bytes = 8 TB of memory!

The use of a kernel density estimator shows a visual evidence of sources and sinks. This evidence can be exploited to reduce the number of clustered particles, as a sampling can be applied according to the probability function. The kernel density estimator provides a probability for each pixel. Dividing each pixel value by the sum of all values results in probabilities to a scale from zero to one considering the probability of each pixel. Afterwards, x random numbers between zero and one were generated (for evaluation of clustering algorithms, x has been set to 5000 as this provides useful results). A lookup of this randomly chosen value results in pixel coordinates where a start respectively end point is set. This method is able to cope with different densities, as the probability of choosing a pixel having a high probability computed by KDE is higher as the range of this pixel is bigger.

Figure 5.3a shows 20 sample pixel densities. Dividing each pixel value by the sum of all values results in unequal probabilities for each pixel, shown in Figure 5.3b. Thus, the randomly generated number has a higher probability to represent a pixel having a high density, as the interval for this pixel is larger than for pixels having small densities.

5.2 Expectation Maximization

The Expectation Maximization algorithm takes the estimated number of clusters as input argument. As one usually does not know the number of clusters, it is very hard to estimate the number of clusters. In combination with the elimination of widely distributed clusters described in Section 3.2.2.2, the choice of the number of clusters is not crucial any more, as only dense clusters should be preserved. The duration needed for the clustering process depends on the number of clusters and varies from one second



(a) number of clusters=5 (b) number of clusters=10

(c) number of clusters=20 (d) number of clusters=30

Figure 5.4: Source clusters obtained by expectation maximization using different estimated number of clusters without applying a threshold T

(five clusters) to 23 seconds (30 clusters). Repeated clustering yields in similar results, thus the detected clusters are relatively stable.

To estimate the number of clusters is not as crucial as it is when using another clustering algorithm, as dense clusters are preserved nearly independently of the estimated numbers of clusters, depicted in Figure 5.4 and 5.5. The main source is modeled accurately at all estimated number of clusters, but a high number of estimated clusters is more likely to find real sources and sinks. Hence, the estimated number of clusters has been set to 20 for the evaluation of different thresholds T introduced in Equation (3.3) to eliminate widely distributed clusters. To choose an appropriate value for α is challenging – Figure 5.6 depicts the influence of α on start points, Figure 5.7 the influence on end points. Expectation Maximization was not able to identify the main source accurately at any threshold T, but sink clusters were identified correctly at all three chosen thresholds. The advantage of Expectation Maximization is the easy choice of input parameters and that it only has minor effects on the results.

These results have been strengthened by the analysis of the second train station video. Figure 5.8 depicts 20 source clusters and the reduced source clusters after applying the threshold, the sink clusters are shown in Figure 5.9. The algorithm does not preserve the main source, but one main exit was preserved during the reduction process correctly

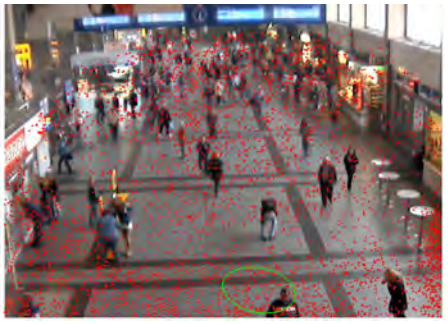
(a) number of clusters=5

(b) number of clusters=10

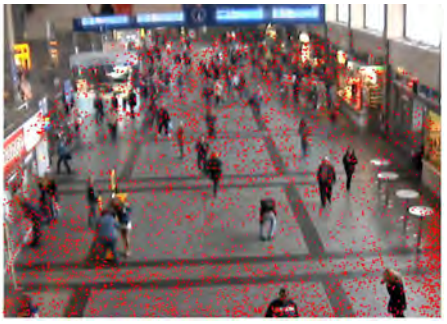
(c) number of clusters=20

(d) number of clusters=30

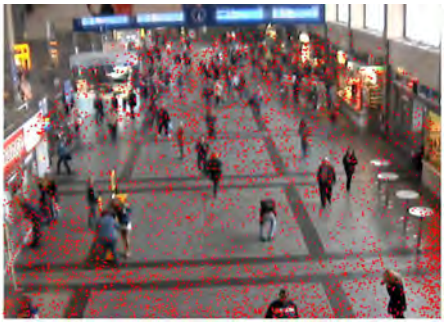
Figure 5.5: Sink clusters obtained by expectation maximization using different estimated number of clusters without applying a threshold T



(a) $\alpha = 0.3$

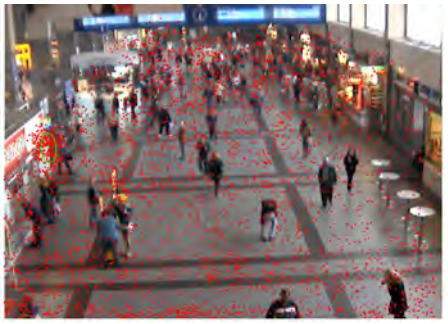


(b) $\alpha = 0.4$

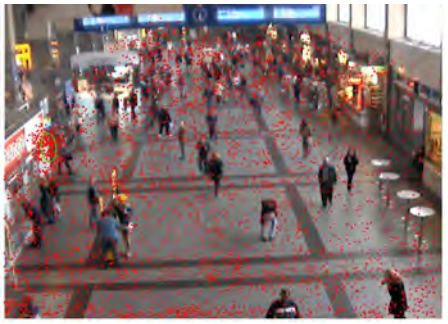


(c) $\alpha = 0.5$

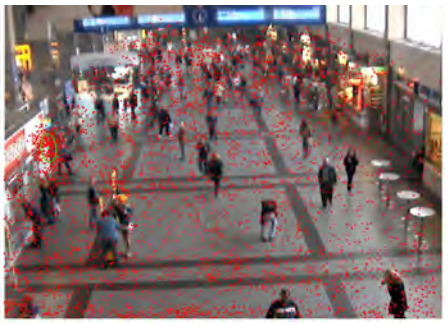
Figure 5.6: Source clusters obtained by expectation maximization using different thresholds defined by α (number of clusters=20)



(a) $\alpha = 0.3$



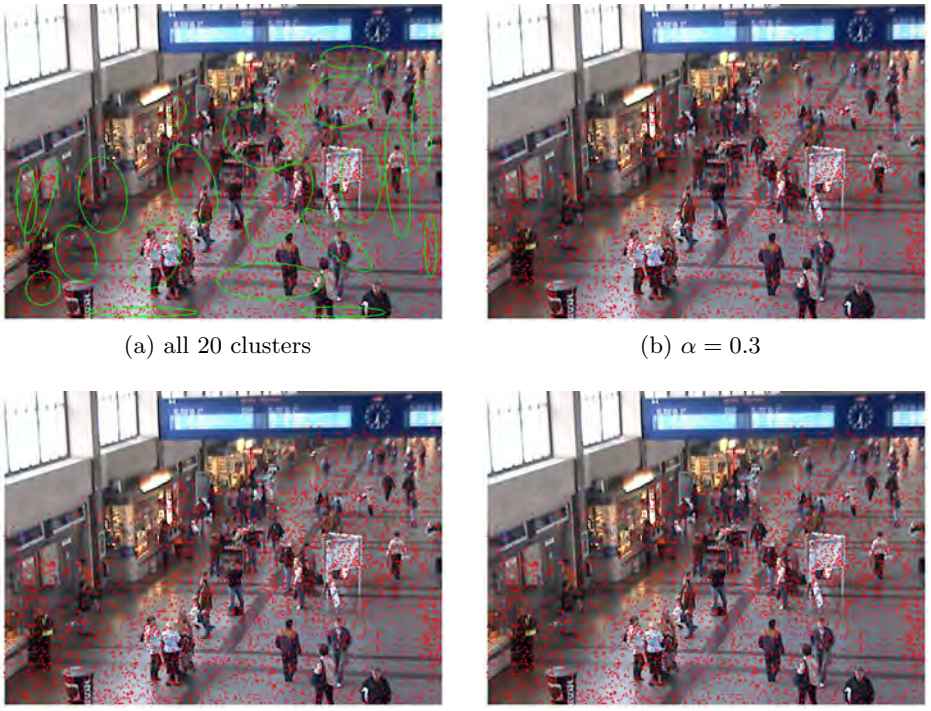
(b) $\alpha = 0.4$



(c) $\alpha = 0.5$

Figure 5.7: Sink clusters obtained by expectation maximization using different thresholds defined by α (number of clusters=20)

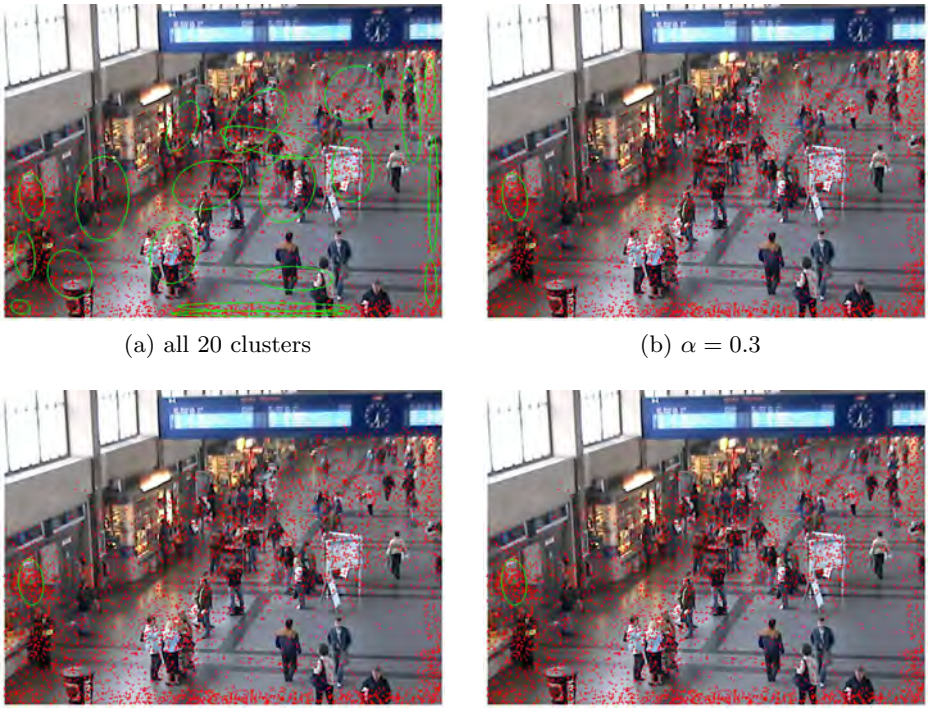
shown in Figure 5.9. Again, the choice of input parameters does not affect the results enormously.



(a) all 20 clusters (b) $\alpha = 0.3$

(c) $\alpha = 0.4$ (d) $\alpha = 0.5$

Figure 5.8: Source clusters obtained by expectation maximization using different thresholds defined by α analyzing the second train station video (number of clusters=20)



(a) all 20 clusters (b) $\alpha = 0.3$

(c) $\alpha = 0.4$ (d) $\alpha = 0.5$

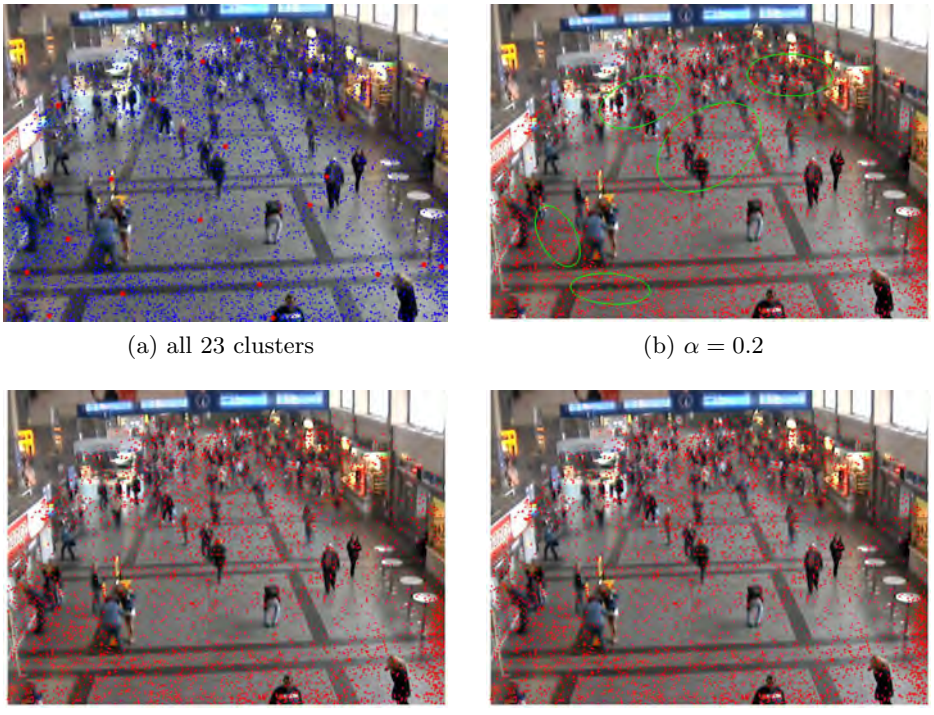
Figure 5.9: Sink clusters obtained by expectation maximization using different thresholds defined by α analyzing the second train station video (number of clusters=20)

5.3 PG-Means

PG-Means is able to detect the number of clusters automatically. Still, there are two input arguments: γ, the probability of making a type-1 error and the number of projections, which should be set in a range of 12-18 according to [20]. The number of projections was set to 15 and γ was set to 0.05. As the detected number of clusters is higher than the number of real sources and sinks, the threshold criterion was applied again. Finding the number of clusters is very time consuming, as the number of clusters is increased iteratively and therefore many computations are done – hence it is not very astonishing that clustering using pg-means took approximately 20 minutes, detecting 23 clusters.

Figure 5.10a and 5.11a depicts all 23 detected clusters. The real source in the lower right corner was not detected accurately shown in Figure 5.10b. The real sink cluster was detected correctly, applying different thresholds α from 0.3 to 0.5. As it took this clustering algorithm very long to determine the clusters and it did not provide better results, other clustering algorithms are preferred.

Applying pg-means to the second train station video, neither the main source nor the main sinks were found by this algorithm. Figure 5.12 depicts the 18 source clusters



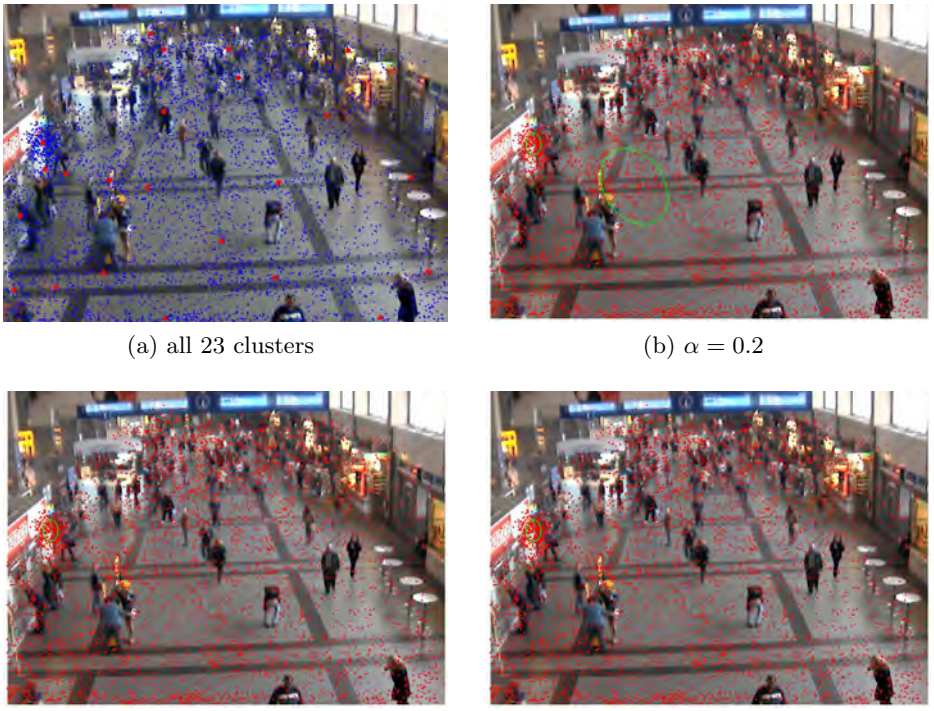
(a) all 23 clusters

(b) $\alpha = 0.2$

(c) $\alpha = 0.3$

(d) $\alpha = 0.4$

Figure 5.10: Source clusters obtained by pg-means applying different thresholds defined by α

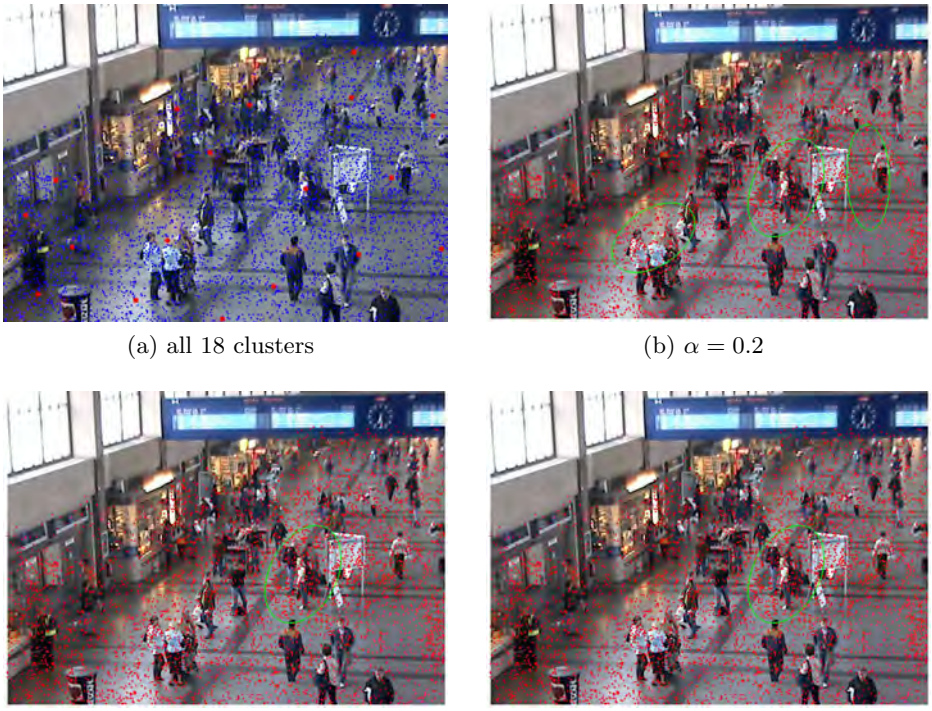


(a) all 23 clusters (b) $\alpha = 0.2$

(c) $\alpha = 0.3$ (d) $\alpha = 0.4$

Figure 5.11: Sink clusters obtained by pg-means applying different thresholds defined by α

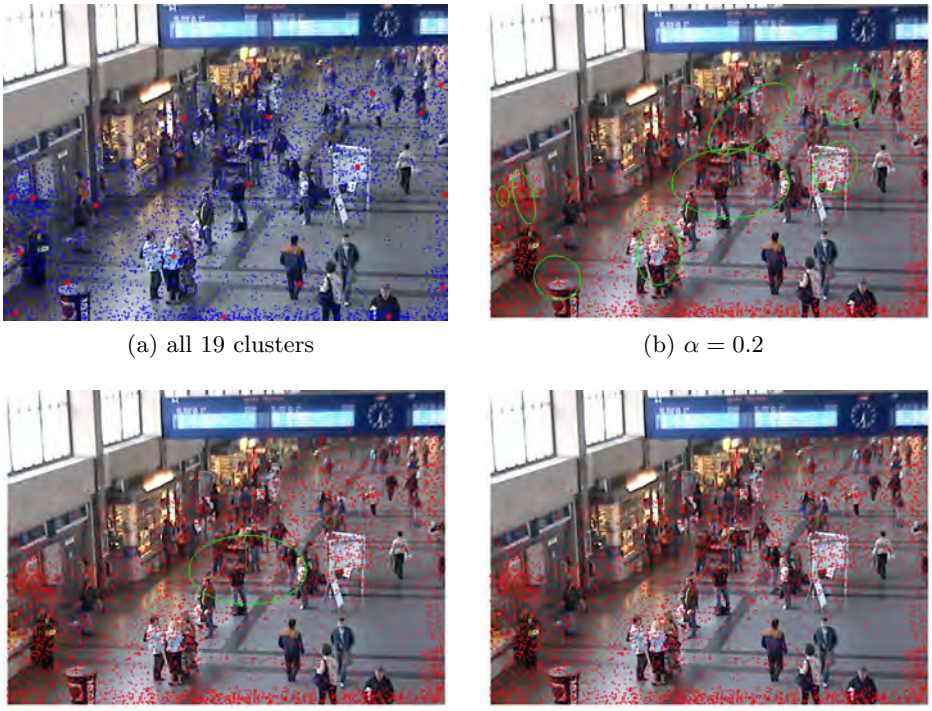
detected by pg-means. The main source was not preserved during the reduction process, as only one cluster in the middle was preserved, not representing a real source. The 19 sink clusters obtained by pg-means are shown in Figure 5.13. Applying a threshold to reduce the number of clusters also results in eliminating the main sinks.



(a) all 18 clusters (b) $\alpha = 0.2$

(c) $\alpha = 0.3$ (d) $\alpha = 0.4$

Figure 5.12: Source clusters obtained by pg-means applying different thresholds defined by α obtained by the second train station video



(a) all 19 clusters (b) $\alpha = 0.2$

(c) $\alpha = 0.3$ (d) $\alpha = 0.4$

Figure 5.13: Sink clusters obtained by pg-means applying different thresholds defined by α obtained by the second train station video

5.4 Mean Shift

Repeated Mean Shift clustering does not yield in the same results and it took the clustering algorithm about 1.5-2 seconds to find the clusters. Mean shift clustering needs a bandwidth specified, defining the radius within points are classified to belong to the same cluster. As sources and sinks are usually represented by small dense clusters, the bandwidth has been set to find many clusters and the number of clusters was reduced afterwards using a threshold α introduced by [15].

Figure 5.14 and 5.15 show the clustering results using different bandwidth values starting with a bandwidth of 0.05 up to a bandwidth of 0.1. Setting a bandwidth of 0.05 results in approximately 100 clusters, whereas a bandwidth of 0.1 results in approximately 15 clusters. Using a bandwidth of 0.1 reduces the number of clusters, thus the primary source and sink are not obtained, as clusters representing the primary source and sink were removed during the reduction process. Hence, a bandwidth of 0.05 was used, as it produces a high number of clusters and reducing the number of clusters yields in better results.

The influence of different thresholds T introduced in Equation 3.3 is shown in Figure 5.16 and 5.17. Setting $\alpha = 0.003$ results in too many sources, whereas a value of 0.005 eliminates all sources. Only a value of 0.004 results in reasonable sources – hence the choice of α is challenging. The sink is more distinctive as the source, thus values from 0.005 to 0.008 preserve the sink while eliminating all other clusters accurately.

Evaluation with another video sequence from the train station shows that mean shift is not able to detect the main source while using a bandwidth of 0.05, depicted in Figure 5.18. Figure 5.19 shows the sensitivity of the algorithm to the chosen value of α - only one main sink is preserved by the algorithm correctly.

(a) bandwidth=0.05

(b) bandwidth=0.07

(c) bandwidth=0.085

(d) bandwidth=0.1

Figure 5.14: Source clusters obtained by mean shift using different values as bandwidth without applying a threshold T

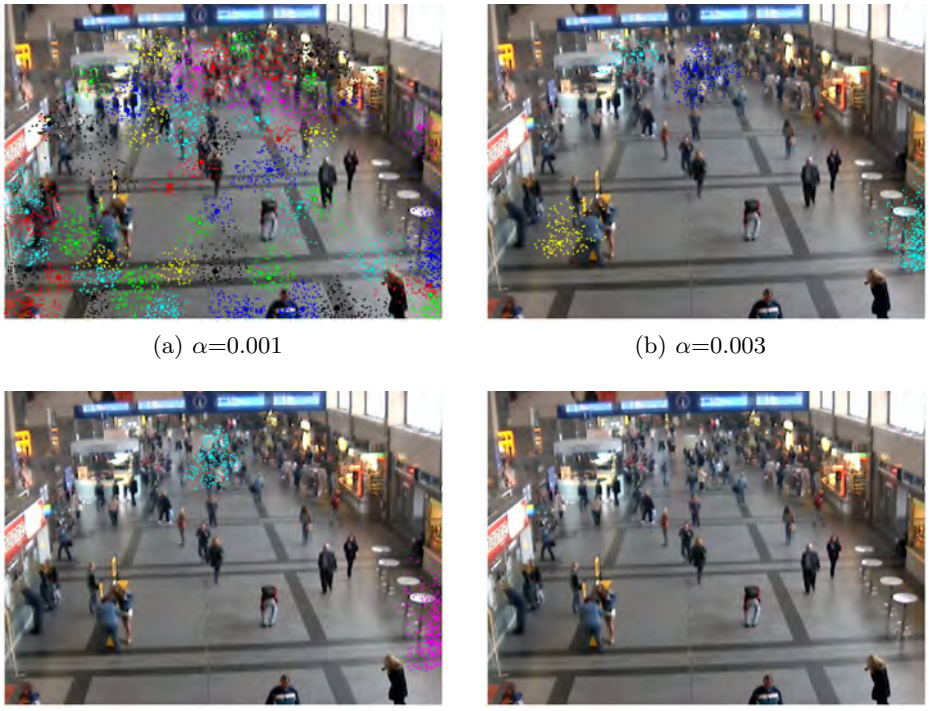
(a) bandwidth=0.05

(b) bandwidth=0.07

(c) bandwidth=0.085

(d) bandwidth=0.1

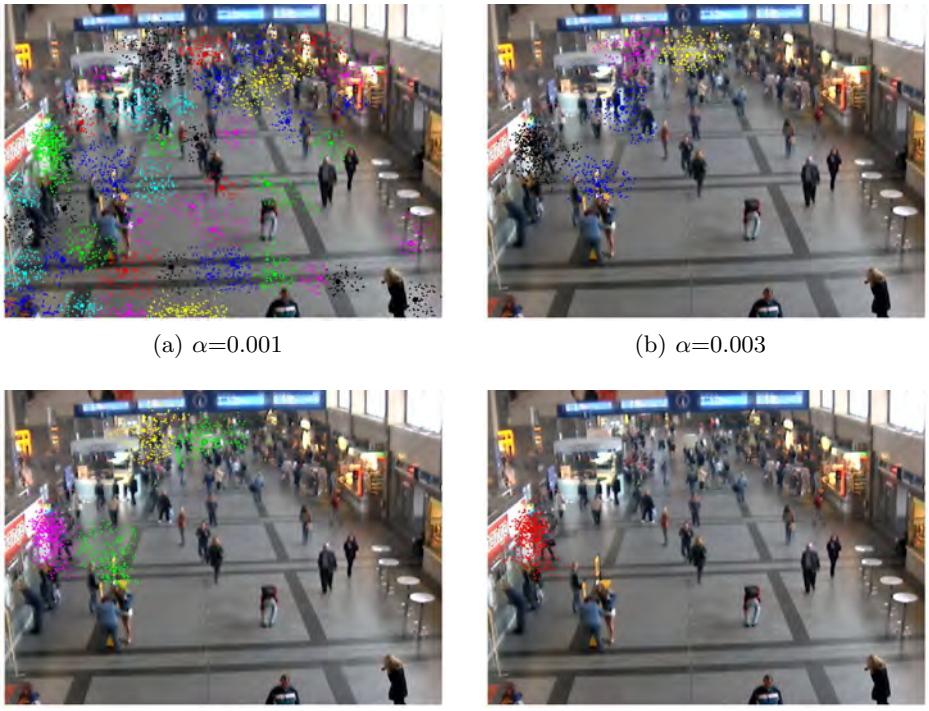
Figure 5.15: Sink clusters obtained by mean shift using different values as bandwidth without applying a threshold T



(a) α=0.001 (b) α=0.003

(c) α=0.004 (d) α=0.005

Figure 5.16: Source clusters obtained by mean shift using different thresholds defined by α (bandwidth=0.05)



(a) α=0.001 (b) α=0.003

(c) α=0.004 (d) α=0.005

Figure 5.17: Source clusters obtained by mean shift using different thresholds defined by α (bandwidth=0.05)

(a) All 100 source clusters

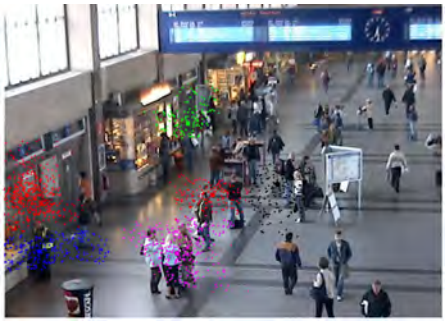
(b) α=0.003

(c) α=0.004

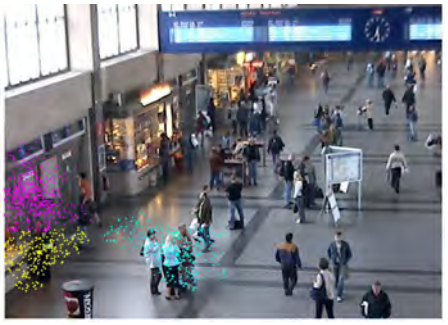
(d) α=0.005

Figure 5.18: Source clusters obtained by mean shift on the second train station video using different thresholds defined by α (bandwidth=0.05)

(a) All 84 sink clusters



(b) α=0.003



(c) α=0.004

(d) α=0.005

Figure 5.19: Sink clusters obtained by mean shift on the second train station video using different thresholds defined by α (bandwidth=0.05)

5.5 DBSCAN

DBSCAN clustering of 5000 points took between 10 and 30 seconds and repeated DBSCAN clustering yielded in exactly the same results. Two parameters can be modified: k, the minimal number of objects considered as a cluster and the threshold α to eliminate widely distributed cluster.

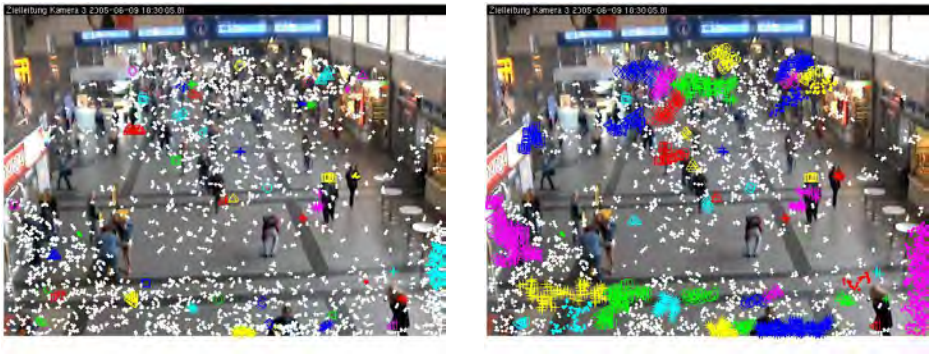
Figure 5.20 and Figure 5.21 depict the influence of k on clustering results. Each cluster is represented by a different marker and color – white points are outliers not assigned to any cluster. Feasible results are obtained using $k <= 3$ – the higher k, the higher the number of points within a cluster. Setting $k = 4$ results in big clusters, not representing the real sources and sinks any more as some clusters are merged.

Using $k = 3$ and applying different thresholds does not yield in desirable results, depicted in Figure 5.22. Applying a threshold with $\alpha = 0.007$ does not find the real sources and sinks. Thus, α was set to 0.005, finding real sources and sinks but not eliminating widely distributed cluster. Hence, there are too many points within each cluster resulting in too big clusters. To reduce the cluster size, k needs to be set to a value smaller than 3. Setting k to 2 results in finding the real sources and sinks even after applying the threshold, depicted in Figure 5.23. The thresholds are very sensitive, hence different thresholds for sources and sinks were applied. The influence of α should be as low as possible, thus k has been set to 1. Figure 5.24 depicts different thresholds applied to clustered data using $k = 1$. Applying a threshold of $\alpha = 0.005$ results in eliminating all sources, but preserving the real sink. As the source is not as distinctive as the sink, it is harder to preserve the source – therefore the choice of α is crucial. Accepting the fact that only very distinctive sources and sinks are preserved, the choice of α is not crucial any more, as the sink was preserved using different values of α up to 0.009.

Analysis of the second train station yields in reasonable results when using $k = 2$ as input parameter. Using this configuration and applying a threshold defined by $\alpha = 0.003$ preserves the main source and the main sinks correctly, as it was not done by any other clustering algorithm. As the source is not as dense as the sinks, the source was eliminated while applying other thresholds in Figure 5.25. The sinks are relatively robust against the choice of α as at least the main sink is preserved depicted in Figure 5.26.

5.6 Self-Tuning Spectral Clustering

Self-tuning spectral clustering takes the number of neighbors to be considered in local scaling as input parameter – but a change of this parameter does not affect the results.



(a) k=1 (b) k=2

(c) k=3 (d) k=4

Figure 5.20: Source clusters obtained by DBSCAN using different values for k

(a) k=1 (b) k=2

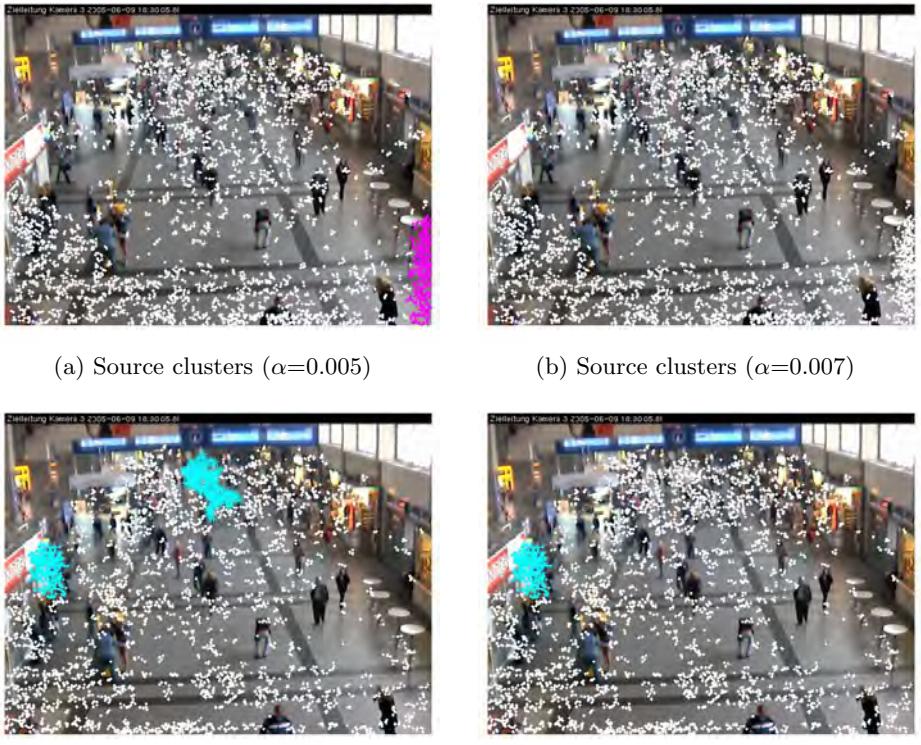
(c) k=3 (d) k=4

Figure 5.21: Sink clusters obtained by DBSCAN using different values for k

(a) Source clusters (α=0.005) (b) Source clusters (α=0.007)

(c) Sink clusters (α=0.007) (d) Sink clusters (α=0.01)

Figure 5.22: Clusters obtained by DBSCAN using different thresholds α with $k = 3$



(a) Source clusters (α=0.005) (b) Source clusters (α=0.007)

(c) Sink clusters (α=0.005) (d) Sink clusters (α=0.007)

Figure 5.23: Clusters obtained by DBSCAN using different thresholds α with $k = 2$

(a) Source clusters (α=0.002)

(b) Source clusters (α=0.005)

(c) Sink clusters (α=0.002)

(d) Sink clusters (α=0.005)

Figure 5.24: Clusters obtained by DBSCAN using different thresholds α with $k = 1$

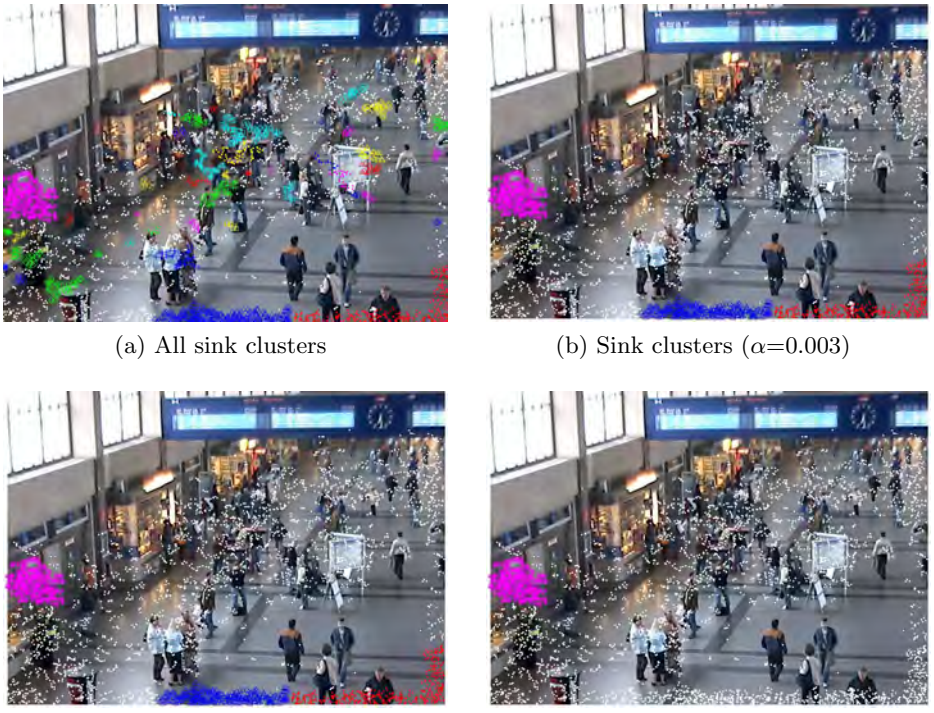
(a) All source clusters

(b) Source clusters (α=0.003)

(c) Source clusters (α=0.005)

(d) Source clusters (α=0.007)

Figure 5.25: Source clusters obtained by DBSCAN applied on the second train station video using different thresholds defined by α with $k = 2$



(a) All sink clusters (b) Sink clusters (α=0.003)

(c) Sink clusters (α=0.005) (d) Sink clusters (α=0.007)

Figure 5.26: Sink clusters obtained by DBSCAN applied on the second train station video using different thresholds defined by α with $k = 2$

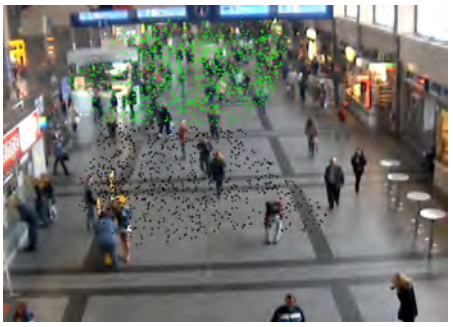
On the other hand, it is very memory and time consuming – clustering 5000 points took approximately 20 minutes, but repeated clustering leads to the same results. As self-tuning spectral clustering tries to find the correct number of clusters but does not eliminate any cluster, the threshold criterion introduced in Equation 3.3 was applied.

This algorithm was not able to detect the source, not even approximately – hence the threshold was not applied to source clusters as it would not yield in desired results. Figure 5.27a shows the source clusters obtained by self-tuning spectral clustering, Figure 5.27b depicts the sink clusters. The threshold criterion was applied to sink cluster, not preserving the main sink, but eliminating it which is shown in Figure 5.27c. Hence, no more different thresholds were applied as no threshold would be able to preserve the real sink and eliminate all other clusters.

Figure 5.28 confirms these results by applying spectral clustering to the second train station video. Again, spectral clustering was not able to detect the main source and the main sinks correctly, as these clusters were not dense enough to be preserved during the reduction process.

(a) Source clusters

(b) Sink clusters



(c) Sink clusters after applying a threshold T defined by $\alpha = 0.02$

Figure 5.27: Self-tuning spectral clustering

5.7 Growing Neural Gas

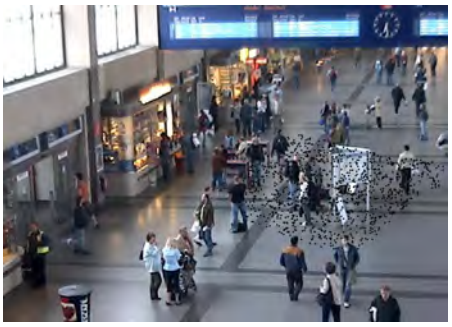
Growing Neural Gas in combination with minimum description length requires over 20 different input parameters and clusters 5000 data points in 15-60 seconds, depending on the number of clusters. Setting these parameters is very challenging and influences the results of GNG dramatically. One does not need to specify the number of clusters directly, but indirectly by setting the number of learning steps and the frequency of inserting new nodes; also the lifetime of edges and corresponding nodes influences the number of nodes.

Figure 5.29a and 5.29b depict the influence of the number of learning steps and the frequency of inserting new nodes on clustering results of source cluster, Figure 5.30a and 5.30b on the sink cluster. Results after applying the MDL reduction criterion are shown in Figure 5.29c and 5.29d, respectively in Figure 5.30c and 5.30d. The influence of edge lifetime is depicted in Figure 5.31.

The choice of the lifetime of edges is not as critical as the choice of the number of nodes. As GNG and MDL require a huge number of input parameters set, it is very difficult to find optimal settings. In our evaluation the optimal settings in terms of eliminating as many nodes as possible and preserving only real sources and sinks, were not found as

(a) Source clusters

(b) Sink clusters



(c) Source clusters after applying a threshold T defined by $\alpha = 0.02$

(d) Sink clusters after applying a threshold T defined by $\alpha = 0.05$

Figure 5.28: Self-tuning spectral clustering - video 2

the number of combinations was too high to evaluate all possible combinations. As the clustering algorithm was not able to represent the main source and sink correctly due to their functionality, no more evaluation on other videos has been done.

(a) high number of nodes

(b) low number of nodes

(c) Reduced source cluster (high number of nodes)

(d) Reduced source cluster (low number of nodes)

Figure 5.29: Source clusters obtained by GNG and MDL

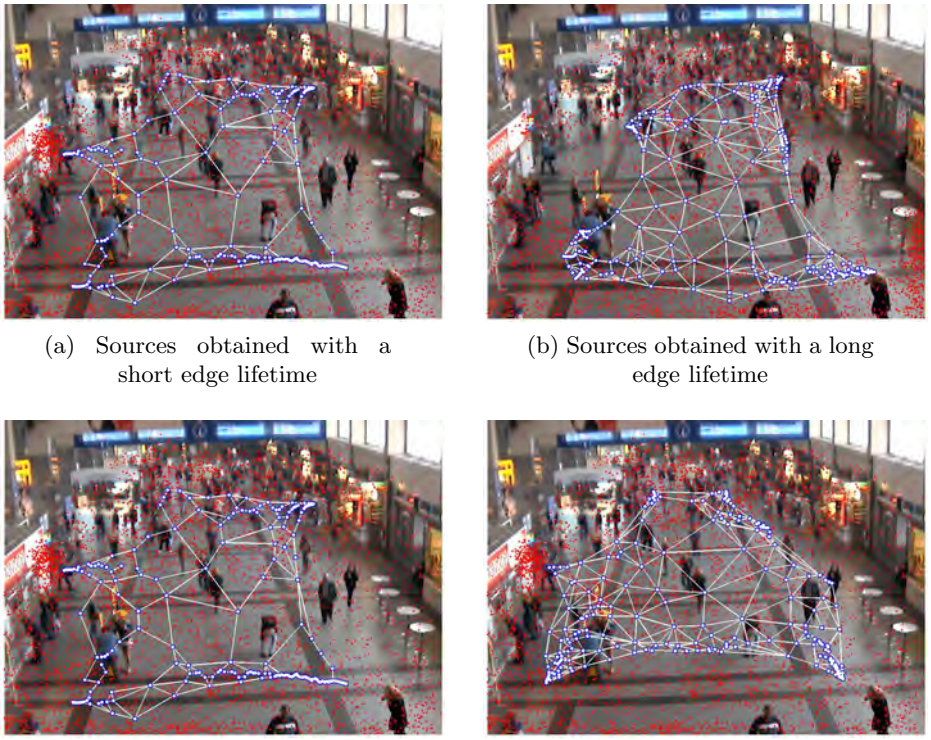
(a) high number of nodes

(b) low number of nodes

(c) Reduced sink cluster (high number of nodes)

(d) Reduced sink cluster (low number of nodes)

Figure 5.30: Sink clusters obtained by GNG and MDL



(a) Sources obtained with a short edge lifetime

(b) Sources obtained with a long edge lifetime

(c) Sinks obtained with a short edge lifetime

(d) Sinks obtained with a long edge lifetime

Figure 5.31: Influence of parameter "edge lifetime" on clustering results

5.8 Comparison of Cluster Algorithm Run Times

As six different clustering algorithms have been evaluated, large run time differences have been noticed. In this Section we summarize these differences. All computations were done on the Intel Core2Quad Q9450 @2.66 GHz using Windows Vista x64 with 4 GB main memory. The run times of evaluated algorithms can be found in Table 5.2.

algorithm	run time
Expectation Maximization	1 - 23 seconds
PG-Means	approximately 20 minutes
Mean Shift	1.5 - 2 seconds
DBSCAN	10 - 30 seconds
Self-Tuning Spectral Clustering	approximately 20 minutes
Growing Neural Gas	15 - 60 seconds

Table 5.2: Run time of cluster algorithms

The evaluation of these clustering algorithms has shown, that PG-Means and Self-Tuning Spectral Clustering had an extremely long run time hence they are not very feasible – the run time of GNG may be acceptable, depending on the parameters. The other clustering algorithms achieved very practicable run times, thus enabling clustering already during the particle advection process.

Chapter 6

Conclusion

Using a particle advection approach instead of individual tracking facilitates the analysis of dense crowded scenes in real-time. Analysis over a longer time period yields reasonable results for obtaining sources and sinks. In our tests we find that using improvements like hierarchical advection or the use of world coordinates improve the quality of particle advection algorithm noticeably, whereas the use of a particle hopping detection mechanism enhances the quality of trajectories but does not improve the overall quality. Depending on the task, experiments have shown that the quality of trajectories or the quality of sources and sinks can be improved using the introduced approaches.

Due to the high number of particles used, clustering is still very challenging but a feasible approach to handle this amount of data was developed by proposing a data reduction mechanism. Reasonable results have been achieved by using the Expectation Maximization and the DBSCAN algorithm. All other algorithms (PG-means, Mean Shift, Self-Tuning Spectral Clustering and Growing Neural Gas) have shown some difficulties, impeding the correct detection of sources and sinks in our test data.

Future work can deal with the problem of choosing an appropriate threshold to preserve clusters and sinks correctly. Another goal would be to implement the clustering process in real-time, to achieve a reasonable combination of the real-time particle advection algorithm and the higher level knowledge derived from clustering.

Bibliography

[1] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, New York, 2006.

[2] D. Comaniciu and P. Meer. Mean shift analysis and applications. In *The Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 2, pages 1197 – 1203, 1999.

[3] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *Proceedings of 2nd International Conference on Knowledge Discovery and Data Mining (KDD-96)*, pages 226 – 231. AAAI Press, 1996.

[4] D. Greene. Graph Partitioning and Spectral Clustering, 2004. URL `http://lab.bcb.iastate.edu/sandbox/pbais05/backup/goodpapers/Greene_MLG04.ppt`. [last accessed September 24, 2009].

[5] L. Zelnik-Manor and P. Perona. Self-Tuning Spectral Clustering. In *Advances in Neural Information Processing Systems*, volume 17, pages 1601 – 1608. MIT Press, 2004.

[6] H. T. Nguyen and A. W. M. Smeulders. Fast occluded object tracking by a robust appearance filter. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(8):1099 – 1104, 2004.

[7] A. Yilmaz, X. Li, and M. Shah. Contour-Based Object Tracking with Occlusion Handling in Video Acquired Using Mobile Cameras. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(11):1531 – 1536, 2004.

[8] C. Gentile, O. Camps, and M. Sznaier. Segmentation for robust tracking in the presence of severe occlusion. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2, 2001.

[9] Z. Tao and N. Ram. Tracking multiple humans in complex situations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(9):1208 – 1221, 2004.

[10] M. Rodriguez, S. Ali, and T. Kanade. Tracking in Unstructured Crowded Scenes. In *International Conference on Computer Vision*, 2009.

[11] D. M. Blei and J. D. Lafferty. Correlated Topic Models. *Annals of Applied Statistics*, 1(1), 2007.

[12] P. Sand and S. Teller. Particle Video: Long-Range Motion Estimation Using Point Trajectories. *International Journal of Computer Vision*, 80(1):72 – 91, 2008.

[13] A. Yilmaz, O. Javed, and M. Shah. Object tracking: A survey. *ACM Computing Surveys*, 38(4):13, 2006.

[14] M. Werlberger, W. Trobin, T. Pock, A. Wedel, D. Cremers, and H. Bischof. Anisotropic Huber-L1 optical flow. In *Proceedings of the British Machine Vision Conference*, London, UK, 2009.

[15] D. Makris and T. Ellis. Learning Semantic Scene Models From Observing Activity in Visual Surveillance. *IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics)*, 35:397 – 408, 2005.

[16] C. Stauffer. Estimating Tracking Sources and Sinks. In *Conference on Computer Vision and Pattern Recognition Workshop*, pages 35 – 35, Madison, Wisconsin, USA, 2003.

[17] X. Wang, K. Tieu, and E. Grimson. Learning Semantic Scene Models by Trajectory Analysis. In *Computer Vision – ECCV 2006*, volume 1, pages 110 – 123, Graz, Austria, 2006.

[18] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, second edition, 2004.

[19] E. Parzen. On estimation of a probability density function and mode. *The Annals of Mathematical Statistics*, 33(3):1065 – 1076, 1962.

[20] Y. Feng and G. Hamerly. PG-Means: Learning the Number of Clusters in Data. *Advances in Neural Information Processing Systems*, 19:393 – 400, 2007.

[21] F. J. Massey Jr. The Kolmogorov-Smirnov Test for Goodness of Fit. *Journal of the American Statistical Association*, 46:68 – 78, 1951.

[22] K. Fukunaga and L. Hostetler. The estimation of the gradient of a density function, with applications in pattern recognition. *IEEE Transactions on Information Theory*, 21(1):32 – 40, 1975.

[23] M. Wertheimer. Untersuchungen zur Lehre von der Gestalt. II. *Psychologische Forschung*, 4:301 – 350, 1923.

[24] U. Von Luxburg. A Tutorial on Spectral Clustering. *Statistics and Computing*, 17: 395 – 416, 2007.

[25] A. Ng, M. I. Jordan, and Y. Weiss. On Spectral Clustering: Analysis and an Algorithm. *Advances in Neural Information Processing Systems*, 14:849 – 856, 2002.

[26] T. M. Martinetz. Competitive Hebbian Learning Rule Forms Perfectly Topology Preserving Maps. In *International Conference on Artificial Neural Networks*, pages 427 – 434, Amsterdam, 1993.

[27] T. M. Martinetz and K. J. Schulten. A "Neural-Gas" Network Learns Topologies. In T. Kohonen, K. Mäkisara, O. Simula, and J. Kangsa, editors, *Artificial Neural Networks*, pages 397 – 402, Amsterdam, 1991.

[28] J. Holmström. Growing Neural Gas. Master's thesis, Uppsala University, Sweden, 2002.

[29] A. Selb. Modellselektion von Clusteringverfahren mittels minimaler Beschreibungslänge. Master's thesis, Vienna University of Technology, Austria, 2000.