CREATION AND CLUSTERING OF TRAJECTORIES FOR FINDING DOMINANT OPTICAL FLOW FIELDS IN CROWDED SCENES

Technical Report

Günther Eibl

Keywords: particle advection, trajectory, crowd, clustering, optical flow.

Abstract: Video footage of real crowded scenes still poses severe challenges for automated surveillance, especially complex flows of crowds have not been properly treated before. This paper shows a method for creating trajectories by advecting particles along the optical flow. These trajectories could be analyzed in many different ways. As an example it is outlined how the resulting trajectories can be used for the determination of dominant flows by subsequent application of a clustering algorithm. The procedure is succuessfully applied to two complex video scenes exhibiting flows that are highly overlapping both in space and time.

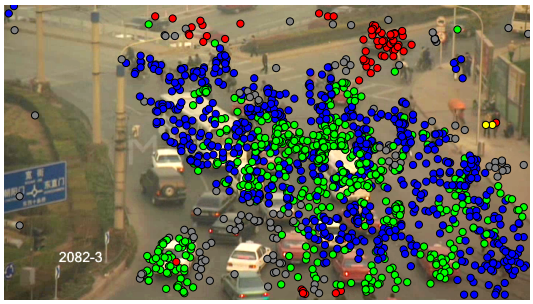
1 INTRODUCTION

Automated visual analysis of scenes involving humans has a wide range of safety and security applications. A model of typical (dominant) people motions within an infrastructure can be an important base for planners, for the detection of abnormal situations and can provide valuable input for realistic pedestrian simulation models. Complex crowded scenes involving many individuals still pose significant challenges for automated analysis. Previous treatments of crowd flows are still limited to rather simple scenarios with only slightly overlapping flows (Ali and Shah, 2007; Andrade et al., 2006; Wright and Pless, 2005; Eibl and Brändle, 2008).

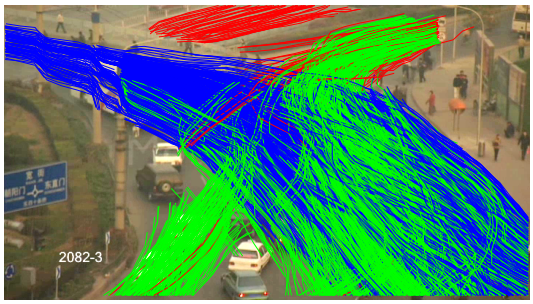
Here we analyze two complex videos which both exhibit both spatially and temporally highly overlapping flows: the video `Traffic` is contained in the publicly available UCF crowd data set (Ali and Shah, 2007), the video `Station` has been recorded by ourselves. Panel (a) of Figure 1 shows the `Traffic` scene including moving cars, bicycles and pedestrians. The four colors denote the four main flow directions (Table 1). The two biggest flows are highly overlapping both in space and time (Figure 1).

Table 1: Coloring scheme

blue	red	yellow	green	gray
←	→	↓	↑	slow particles



(a) Advected Particles

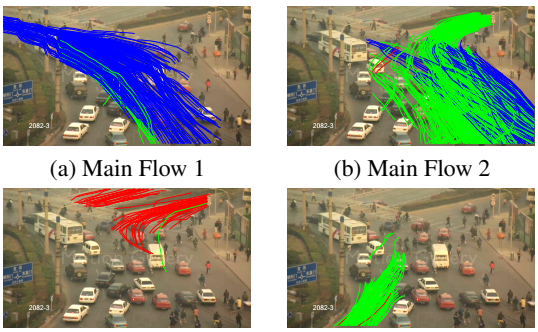


(b) All Trajectories

Figure 1: `Traffic`

Tracer particles are put into the optical flow and flow along the *current* optical flow until they exit the

image (Figure 1 (a)). As a result, each particle leads to a trajectory (Figure 1 (b)). Instead of directly clustering the flow vectors as in (Eibl and Brändle, 2008) we create and cluster these trajectories. In this paper we focus on how these trajectories are properly created. The trajectories could be analyzed in many different ways. Here we outline how they can be used for the determination of dominant flows (Figure 2) by subsequent application of a spectral clustering algorithm (Zelnik-Manor and Perona, 2004).



(a) Main Flow 1 (b) Main Flow 2

(c) Main Flow 3 (d) Main Flow 4

Figure 2: Main flows of `Traffic`

An approach inspired by particle dynamics has already presented before (Ali and Shah, 2007). There the advected particles are used for the determination of flow boundaries which then serve as input for a graph-cut based image segmentation procedure. The image sequences analyzed there do not contain overlapping motions for a region like in Figure 1.

This paper is organized as follows: Section 2 shows how particles are moved (Section 2.1), inserted (Section 2.2), moved backwards (Section 2.3) and describes properties of the advection method (Section 2.4). Section 3 then outlines how the resulting trajectories are clustered into dominant flows.

2 TRAJECTORY CREATION BY PARTICLE ADVECTION

The basic input for the method of particle advection is the optical flow calculated by a state-of the art method (Zach et al., 2007). The result is an optical flow field $\mathbf{u}_f(\tilde{\mathbf{x}}) = \mathbf{u}_f(\tilde{x},\tilde{y}) = (u(\tilde{x},\tilde{y}),v(\tilde{x},\tilde{y}))$ storing the horizontal and vertical velocities u and v at a grid $\tilde{\mathbf{x}} = (\tilde{x},\tilde{y})$ of integer horizontal and vertical positions for each transition from one frame f of the input video to the next frame $f+1$. The resulting flow fields are stored in the form of individual pcm-files. The resulting amount of data can be huge: the 12000 optical flow frames of the scene `Station` need 27.3 GB of memory. Consequently, loading these flows is time consuming and turned out to be one of the main factors influencing computation time. In the design of the particle advection program one of the main goals was to load as few flow frames as possible.

The proposed procedure generates particle trajectories using the optical flow fields. The idea is analogous to putting a swimming leave into a river and recording its position as it flows along the river. Particles p are inserted into the system at position $\mathbf{x}$ of frame f and flow along the current optical flow field $\mathbf{u}_f(\mathbf{x})$. In the next iteration the particles flow along the optical flow field $\mathbf{u}_{f+1}$. This procedure continues until the particle leaves the system or gets stranded. The positions of the particles get stored at each frame f resulting in a trajectory. In contrast to many usual approaches here a particle does not describe a person or a specific feature of the image which gets tracked.

2.1 Forwards Movement

A particle p is described by position, velocity and frame number. The position of a particle at frame f is described by continuously defined numbers $\mathbf{x} = (x,y)$. It is important to define the position continuously, otherwise particles with a speed below 0.5 pixels/frame would never move forward. Since the velocity is only known at a grid, for moving the particle at a continuously defined position $\mathbf{x}$, first the nearest point on the grid $\tilde{\mathbf{x}}$ is obtained rounding both x and y to the nearest integer values. Then the velocity $\mathbf{u}(\mathbf{x})$ is found as $\mathbf{u}(\tilde{\mathbf{x}})$. In principle, this rounding step is the simplest possible interpolation of the velocity from the grid to the current position of the particle which could be improved using interpolation methods of higher order. However, the flow field provided by the optical flow program is already smoothed in space making a more accurate interpolation method unnecessary. A particle p at frame f is then represented as

$$t_f = (\mathbf{x}, \mathbf{u}(\mathbf{x})). \tag{1}$$

Now we describe how a particle p is moved one frame forward. Using a tiny adaptation the same procedure will also be applied for backwards movement. First the new tentative position $\mathbf{X} = (X,Y)$ is calculated by adding the velocity to the position

$$\mathbf{X} = \mathbf{x} + \mathbf{u}(\mathbf{x}). \tag{2}$$

Then it is checked if the particle is still moving actively. There are three possibilities how the movement of a particle can be terminated

- A particle leaves the system
- A particle gets stranded

- A particle reaches the maximum lifetime

If either of the three conditions holds the status variable moving of particle p is set to false. The first condition checks, if a particle moves out of the current video frame.

A particle is considered as stranded if it does not move further than ε within a given number of frames Δf. In the insertion process particles may be placed at places with no current motion which is one of the main sources of stranded particles. A typical example would set ε to one pixel and Δf to 10. Removal of stranded particles has two benefits: First, the removal of stranded particles frees memory. More importantly, stranded particles can get into several different flows resulting in trajectories which do not occur in reality.

The third condition is a safety condition which should be applied if the other two conditions fail. The lifetime of a particle p is the difference between the actual frame number and the frame number where the particle has been inserted $f(p) - f_{in}(p)$. The maximum lifetime $\Delta f_{\max}$ can easily be estimated by visual inspection of the first part of the video.

For the particles which are still moving (status moving is true) the new position is set to the tentative position $\mathbf{X}$, the new velocity is calculated at frame $f+1$ and the whole procedure continues. After particle p has stopped its movement at frame $f_{out}(p)$ we can represent its forward trajectory $T_{forw}(p)$ by

$$T_{forw}(p) = \{t_f \mid f = f_{in}(p), \ldots, f_{out}(p)\} \quad (3)$$

2.2 Particle Insertion

Particles are seeded uniformly into the part of the current frame where the speed $R = \|\mathbf{u}\|$ exceeds a threshold $R_{\max}$. This threshold is not defined as an explicit parameter but rather defined as the $1-\alpha$-quantile of all the speed values R occurring at the current frame f. This parameter α can be interpreted as the fraction of the area of the frame where motions occur. Setting α to one results in random uniform particle input into the whole frame.

Particles are continuously inserted into the system. More exactly, N_{ins} particles are inserted each Δf_{ins}'th frame. Since the current frame is used for selecting the insertion area newly developing flows can be followed naturally. On the other hand particles are only set in a fraction α of the frame area which prohibits the insertion of particles into parts of the image showing permanently non-moving objects like walls. As an effect less particles must be discarded in later steps compared to completely uniform insertion ($\alpha = 1$). Thus, less memory is needed for achieving the same number of valid trajectories.

2.3 Backwards Movement

Consider a particle injected in the middle of the system. Forward advection follows the flow only up to its destination so it will be unknown where the particle entered the system. This problem is addressed by letting particle p not only flow forward in time to its destination but also backwards in time to its origin. The whole path $T(p)$ of particle p then consists of the combined backward path $T_{backw}(p)$ and forward path $T_{forw}(p)$.

In principle the full video could be processed in forward direction yielding all the forward paths $T_{forw}(p)$ followed by a backwards sweep yielding the backward paths $T_{backw}(p)$ of all particles p. Since the final result would only be available at the end of the whole video, a method involving blockwise forward and backwards sweeps yielding results after F frames was developed (Figure 3). Note that due to memory limitations at each single step only one frame is loaded at each single step.

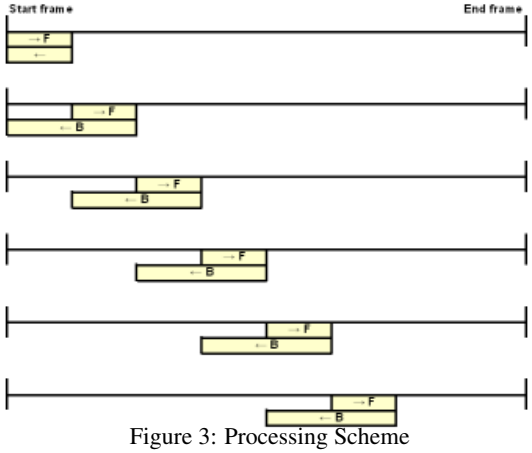


Figure 3: Processing Scheme

First, particles are moved forward for F frames which are loaded one at a time. Then the particles are split into two sets. The first set P^+ consists of the particles which are still moving in forward direction (status variable moving is true). These particles are kept in memory and will continue their forward movement later. The second set P^- consists of particles which stopped moving in forward direction. Thus, the destination of these particles is known but not the origin. For later use the forward part $\{T_{forw}(p|p \in P^-)\}$ of their trajectories must be saved. Now all the particles in the set P^- are moved backwards until their origin is known.

This means that each particle of the set P^- is put at the location $\mathbf{x} = (x_{in}, y_{in})$ of frame f_{ins} where it has been originally inserted and moves back updating

its position with the negative velocity of the current frame

$$\mathbf{X} = \mathbf{x} - \mathbf{u}(\mathbf{x}). \tag{4}$$

As for the forward movement it is checked if the particle is still moving actively. The positions of the moving particles are updated and the velocities of the next frame $f-1$ are again chosen as taking the flow field $\mathbf{u}_{f-1}$ at the nearest position. So backwards movement is done in exactly the same manner as the forward movement except that the frame number decreases, the sign of the velocity is reversed for the position update and only the particles of P^- are inserted (no continuous particle insertion).

After B backwards movements, also the backward trajectories of the particles in P^- are known and the forward and backward trajectories can be combined. So after every F frames (counted in forward direction) new complete trajectories are available. The choice $\mathrm{B} = \mathrm{F} + \Delta f_{\max}$ ensures that backwards movement has stopped for all particles in the set P^-. A reasonable choice for F is $\Delta f_{\max}$, so our choice for F and B is

$$\mathrm{F} = \Delta f_{\max}, \quad \mathrm{B} = 2\Delta f_{\max}. \tag{5}$$

After the backwards block has been completed the particles in the set P^+ continue their movement. Note that due to the continuous insertion of particles in forward direction the system does not run out of particles. The number of particles flowing in forward direction is given by the balance of influx (determined by the ratio $N_{ins}/\Delta f_{ins}$) and the outflux which is determined by the video itself (size and velocities). Using setting (5) each frame is loaded 3 times.

2.4 Properties of Particle Advection

Observing the original video data with flowing particles overlaid on it is the most suitable method for the analysis of the correctness of the flow and the investigation of occurring phenomena. Snapshots of such videos are shown in Figures 1(a) and 4. Here we describe several phenomena we have seen in the videos of the advecting particles.

Particles are predominantly placed on moving objects. Persons far away from the camera appear smaller, therefore fewer particles are placed on them and the number of particles per person is not constant. The insertion parameters (more precisely $N_{ins}/\Delta f_{ins}$ and also α) should be set such that all moving objects contain particles (Figures 1(a) and 4). Increasing the number of particles further does not improve results, because the number of moving objects in a scene is finite.

Particles typically move along with the objects they were initially placed upon. However, particles

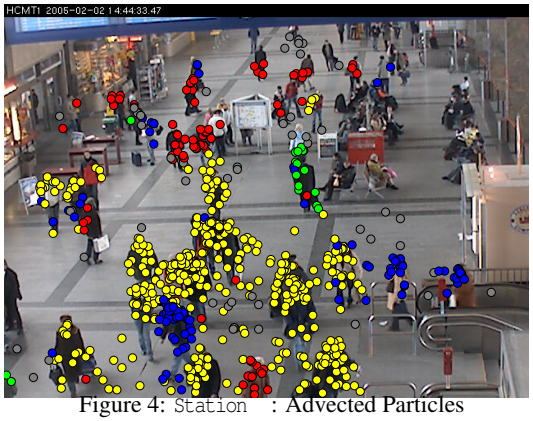


Figure 4: Station : Advected Particles

can be "lost" resulting in stranded particles. This phenomenon predominantly occurs for persons appearing small in the video. The resulting trajectories are then only sub-trajectories of the whole trajectory. More important, due to the spatial smoothing of the optical flow such lost particles can still have a low velocity and do not get detected as stranded. If such a lost particle is then "picked up" by another person (i.e. if it gets into the flow created by this person) its trajectory is invalid because it does not occur in reality. If the two involved persons travel in different directions the artificial trajectory will exhibit a sharp change of the flow direction enabling the detection of such artificial trajectories.

Some properties are a direct consequence of the fact that the particle advection method is based on the optical flow. At the location of non-moving objects the flow speed is very small. Due to the chosen particle insertion procedure no particles are placed on these objects (Figure 1 (a), stuck cars in the lower left part). Suppose that an object enters, stops, continues moving and exits. Particles are inserted at the objects position and follow it until the object stops. Since they don't move further they will be detected as stranded, then they will be followed backwards to the origin and the trajectory is saved. When the objects starts moving again, new particles are placed on it and follow it first in forward direction to the exit and then backwards to the stopping points resulting in a second trajectory. Thus, for objects moving with intermediate stops the whole trajectory is split into sub-trajectories.

Objects moving behind another object are hidden from the camera and therefore do not contribute to the optical flow. Thus, at certain places no flows occur (e.g. the information sign in the middle or the escalator in the lower right part of Figure 4). Consider persons moving behind such objects. First the person is not hidden so particles are placed on them

and move along its trajectory. When a particle reaches the non-moving obstacle it can not move further, gets detected as stranded and a sub-trajectory of the whole trajectory will be saved (Figure 5). Note that in this example some particles are also lost at stationary objects in the *background* due to the spatial smoothing of the optical flow.

Figure 5: Objects in the Foreground

Due to the smoothing of the optical flow the particles reaching an obstacle in the foreground do not necessarily get stuck because they can still get some velocity from flow in the neighborhood. This situation occurs at the street lamps in the upper left part for the Traffic video.

In the special case where the object in the foreground is another moving person, the situation is more complicated and also depends on the details of the movement. E.g. in Figure 6 due to the sidewards movement of the bag (middle picture) not all particles of the person in the background hop to the person in the foreground.

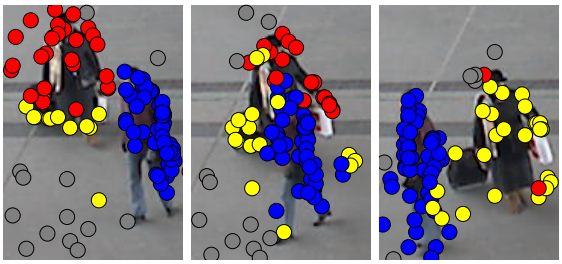


Figure 6: Particle hopping

2.5 Postprocessing of Trajectories

In the first step of postprocessing the length of the trajectories is computed as the sum of the distances between subsequent points and trajectories shorter than a given number of pixels are removed. The remaining trajectories are smoothed using cubic smoothing splines and points are equidistantly (with Δl fixed) resampled along the arclength l of the smoothed trajectory (Bauer et al., 2006). The velocity is newly computed as the space difference between subsequent points and the parametrization along the arclength l of the curve is computed as fifth coordinate. We denote such resampled trajectories of a particle p by

$$Z(p) = (z_k)_{k=1,\ldots,N_k}, \quad \text{for} \quad z_k = (x_k, y_k, u_k, v_k, l_k) \quad (6)$$

Artificial trajectories exhibiting sharp curves are removed by only keeping trajectories for which the direction between subsequent velocity vectors does not extend a given threshold. We have set our threshold to 57 degrees based on a manually annotated subset of trajectories of the `Station` trajectories. The resulting trajectories are shown in Figures 7 and 1 (b).

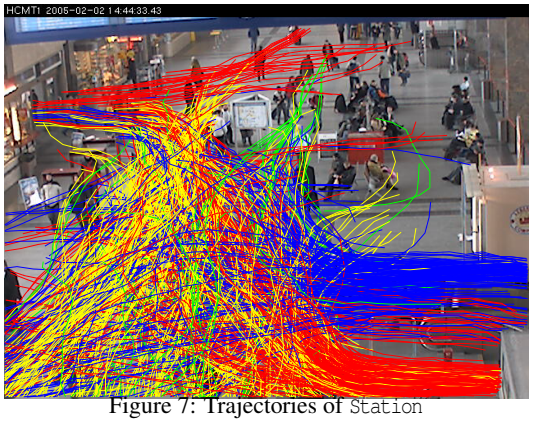


Figure 7: Trajectories of `Station`

3 CLUSTERING INTO DOMINANT FLOWS

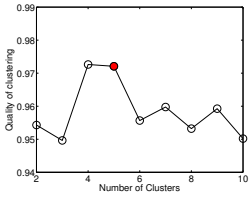
In this section we show one of many possible ways how the resulting set of trajectories can be clustered into dominant flows. For clustering we chose a random subset of 1000 trajectories, using more trajectories did not change results. Prior to clustering the horizontal positions and velocities are divided by the number of columns and the vertical positions and velocities are divided by the number of rows. Afterwards the modulus of the velocities is normalized to one. This treatment ensures that angles are computed the same way for position and velocity coordinates. The arclength coordinate is normalized to the interval [0,1].

The input for subsequent spectral clustering is the distance matrix D of pairwise distances $D^{i,j}$ between the resampled trajectories $Z^i = Z(p^i)$ and $Z^j = Z(p^j)$ of particles p^i and p^j. We take M equidistant points

of trajectory Z^i, compute the minimum distances to the whole trajectory Z^j and sum them up

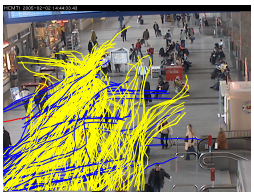
$$D^{ij} = \sum_{m=1,\ldots,M} \left(\min_{k=1,\ldots,N_k} \|z_m^i - z_k^j\|_1 \right). \qquad (7)$$

We symmetrize the distance matrix by computing only the upper part $j > i$ and copying it to the lower part. For both datasets we set $M = 4$. Thus, the first trajectory consists of the source and the origin and two intermediate points.



(a) Number of Clusters

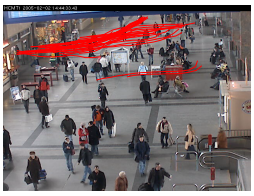
(b) Main Flow 1



(c) Main Flow 2

(d) Main Flow 3

(e) Main Flow 4



(f) Main Flow 5

Figure 8: Main Flows for `Station`

For clustering we used a spectral clustering algorithm (Zelnik-Manor and Perona, 2004) which uses a local scale for the creation of the similarity matrix and additionally provides a method for the estimation of the number of clusters. For both datasets we would have chosen the same number of clusters by visual inspection as this estimation method (Figure 8 (a), result for Traffic not shown).

The clustering results are shown in Figure 8(b)-(f) and Figure 2. Note again that the colors indicate the coarse main direction and not the cluster membership. The last cluster in Figure 2(d) consists of trajectories which stop in the middle of the road due to the shortness of the video and not due to incomplete advection of particles.

Although we did not put much effort into the clustering procedure we obtained reasonable clusters. These two datasets are not tractable by a pure clustering approach (Eibl and Brändle, 2008). To our knowledge there is no other existing approach treating such complex flow video data.

4 CONCLUSIONS

In this paper we showed how trajectories can be created by advecting particles along the current optical flow. We outlined how these trajectories can be clustered into meaningful dominant flows. To our knowledge there is no other existing approach treating such complex flow video data.

Since many properties of the trajectories are a direct consequence of the optical flow there is more room for improvement in the postprocessing than in the advection part itself. This could range from improving the simple method of detecting unproper trajectories up to combining short subtrajectories into longer ones or dividing unproper trajectories into proper subtrajectories.

We also plan to analyze the trajectories in many different ways in order to estimate the density, describe time-dependent phenomena, detect outliers or learn sources and sinks.

REFERENCES

Ali, S. and Shah, M. (2007). A Lagrangian Particle Dynamics Approach for Crowd Flow Simulation and Stability Analysis. In *Proceedings CVPR*.

Andrade, E., Blunsden, S., and Fisher, R. (2006). Modelling Crowd Scenes for Event Detection. In *Proceedings ICPR*.

Bauer, D., Brändle, N., and Seer, S. (2006). Finding highly frequented paths in video sequences. In *Proceedings of the ICPR2006 Conference*, Hongkong.

Eibl, G. and Brändle, N. (2008). Evaluation of clustering methods for finding dominant optical flow fields in crowded scenes. In *Proceedings of the ICPR2008 Conference*, Tampa Bay.

Wright, J. and Pless, R. (2005). Analysis of Persistent Motion Patterns Using the 3D Structure Tensor. In *WACV/MOTION*.

Zach, C., Pock, T., and Bischof, H. (2007). A Duality Based Approach for Realtime TV-l^1 Optical Flow. In *Proc. 29th DAGM Symposium on Pattern Recognition*.

Zelnik-Manor, L. and Perona, P. (2004). Self-Tuning Spectral Clustering. In *Adv. Neural Inf. Process. Syst.*, pages 1601–1608.