Meta-Tracking for Video Scene Understanding
Pierre-Marc Jodoin, Yannick Benezeth, Yi Wang

To cite this version:
Pierre-Marc Jodoin, Yannick Benezeth, Yi Wang. Meta-Tracking for Video Scene Understanding. IEEE International Conference on Advanced Video and Signal-Based Surveillance (AVSS), Aug 2013, Poland. hal-00857916

HAL Id: hal-00857916
https://u-bourgogne.hal.science/hal-00857916
Submitted on 10 Oct 2013

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Meta-Tracking for Video Scene Understanding

Pierre-Marc Jodoin†
Yannick Benezeth††
Yi Wang†
† Université de Sherbrooke
2500 Boul de l’Université, Canada
[PIerre-Marc.Jodoin;Yi.Wang]@usherbrooke.ca
yannick.benezeth@u-bourgogne.fr

Université de Bourgogne
21000 Dijon cedex France

Abstract
This paper presents a novel method to extract dominant motion patterns (MPs) and the main entry/exit areas from a surveillance video. The method first computes motion histograms for each pixel and then converts it into orientation distribution functions (ODFs). Given these ODFs, a novel particle meta-tracking procedure is launched which produces meta-tracks, i.e. particle trajectories. As opposed to conventional tracking which focuses on individual moving objects, meta-tracking uses particles to follow the dominant flow of the traffic. In a last step, a novel method is used to simultaneously identify the main entry/exit areas and recover the predominant MPs.

The meta-tracking procedure is a unique way to connect low-level motion features to long-range MPs. This kind of tracking is inspired by brain fiber tractography which has long been used to find dominant connections in the brain. Our method is fast, simple to implement, and works both on sparse and extremely crowded scenes. It also works on highly structured scenes (highways, traffic-light corners, etc.) as well as on chaotic scenes.

1. Introduction
Surveillance video camera networks are increasingly ubiquitous. As a result, the need for intelligent surveillance systems is becoming a glaring economical issue. Intelligent surveillance systems often focus on anomaly detection [21] (detecting illegal U-turns, locating people in restricted areas, etc.) and scene monitoring [17] (people/car counting, traffic jam detection, etc.). In some cases, the system has a clear definition of what it is looking for (e.g. finding people in restricted areas, cars running against traffic, etc.). However, in other cases, the system has to learn the content of the scene while the video streams in. In general, scene understanding means either learning the scene layout (locating roads, building, bridges, etc.) [3], the scene status (traffic light, congestion, crowd behaviors, etc.) [17] or predominant motion patterns (MPs) [19].

It is widely accepted that grasping the content of a scene is not equally easy for all scenarios. Due to their content, some videos are challenging and require complex solutions. Common issues reported are the following:

1. Clutter: the number of moving objects in a video can lead to challenging situations. That is true in crowds and in heavy traffic jams where several moving persons/vehicles partly occlude other moving objects (see Fig. 1(a)). These scenarios make it difficult (if not impossible) to track every moving object.

2. Structure of the flow: the nature of the flow is another critical variable. In some sequences, the flow is highly structured with objects following well-defined patterns. An example is shown in Fig. 1(b) in which vehicles all run at the same speed without changing lanes. In other scenarios, the flow is chaotic with persons and vehicles crossing each other and heading towards different directions. This is true in shopping malls, airports, and where street traffic is poorly regulated as in Fig. 1(c).

3. Layout: the layout of the scene determines heavily the path of the flow. A simple layout might involve one or two predominant directions (see Fig. 1(b)), but more complicated scenes include splitting and merging of non-linear MPs (roundabouts, crossroad, Y-shape road as in Fig. 1(d)). Recovering and representing such non-linear and long-range MPs (especially when they cross each other) is a difficult task.

In this paper, we present a method to recover MPs that works on sparse scenes as well as on crowded scenes, on structured and chaotic scenes, and can handle non-linear (and yet long-range) MPs. To do so, our method relies on pixel-based orientation distribution functions (ODFs) computed from motion histograms. These ODFs are used to recover meta-tracks, i.e. trajectories of particles transported along the ODFs. Meta-tracks are then clustered to recover predominant MPs and the main entry/exit points.

Our method is inspired by MRI brain tractography [13] which recovers long-range, intricated and cluttered connections between different areas of the brain.

2. Previous Work
Since activities in public areas often follow basic rules, several scene understanding methods quantify space and time into a number of states with transition probabilities. Common models are HMMs [12], Bayesian networks [6], context free grammars [18], and other graphical models [16]. Methods focusing on global behavior understand-
Motion flow vectors are clustered to-

gether with a hierarchical clustering based on a geodesic
distance matrix. As for [10], this method was not meant to
handle crossing and overlapping MPs.

Let us also mention work by Wang et al. [19] which de-
composes the video into clips in which local motion is quan-
tized into words. These words are then clustered into topics,
each clip being modeled as a distribution over these topics.
Although these topics are motion patterns close to the ones
we are looking for, since this technique works only on re-
current activities, it is ill-suited for unstructured flow (as in
Fig. 1(c)). Similar approaches can be found in [9, 12].

3. Our Method

As shown in Fig. 2, our method is a 4-step pipeline
which: 1) computes a motion histogram for each pixel, 2)
converts motion histograms into ODFs, 3) performs meta-
tracking and 4) clusters meta-tracks to recover MPs as well
as the entry/exit points.

Figure 3. Motion histograms showing (A) horizontal, (B) diagonal
right, and (C) bimodal motion (upper left and upper right).

3.1. Motion Histograms

Consider $I$, a video with constant inter-frame interval,
and $(u_t, v_t)$ the optical flow at time $t$ computed by a motion
estimation method. In this work, we use the Horn/Schunk -
Lucas/Kanade method proposed by Bruhn et al. [5] which we found to be a good compromise between precision and speed. Although ODFs (to be defined later) could be obtained with tracklets as in [23], we empirically found that optical flow is much easier and faster to compute. Once optical flow has been computed for each frame, a motion histogram \( M_p(u,v) \) is computed at each pixel \( p \). This histogram contains the number of times a pixel \( p \) had its motion vector \((u_{t,p}, v_{t,p})\) equal to \((u, v)\) during the entire video. Since \( M_p \), takes integer indices, motion vectors \((u_{t,p}, v_{t,p})\) are rounded up to the nearest integer. Fig. 3 shows three pixel-based motion histograms.

### 3.2. Converting Motion Histograms into ODFs

Motion histograms are ill-suited for meta-tracking. Our tracking technique relies on orientation sampling based on an opening angle (to be define in Section 3.3) and computing it on a motion histogram is not only time consuming but also error prone due to aliasing problems. Furthermore, our tracking method is only concerned with flow orientation and not flow amplitude.

An ODF is the probability density function of finding a given motion orientation. As opposed to motion histograms, ODFs have no information on the magnitude of the flow. This makes the ODF representation simpler (1D instead of 2D) which is a key advantage computational wise for the upcoming meta-tracking algorithm. A pixel-based ODF \( \Phi_p(\theta) \) is computed through radial integration:

\[
\Phi_p(\theta) = \frac{1}{\lambda_p} \int_0^R M_p(\theta, r) dr
\]

where \( \lambda_p \) is a normalizing constant and \( M_p(\theta, r) \) a motion histogram in polar coordinates (entry \((u, v)\) is represented by an angle \( \theta \in [0, 360] \) and a distance to the origin \( r > 0 \)).

### 3.3. Meta-Tracking

Once every pixel has been assigned to an ODF, the meta-tracking function can be launched. This procedure lays a grid of particles and has them advected in the scene following an iterative algorithm. Let \( q \) be a particle at position \( \rho_q = (i, j) \) and iteration \( \tau \). At each iteration, the goal is to estimate the next particle position \( \rho_{q+1}^{\tau+1} \) given the ODF \( \Phi_q(\theta) \). Mathematically, this boils down to

\[
\rho_{q+1}^{\tau+1} = \rho_q^\tau + \delta \vec{v}_{q+1}^{\tau+1}
\]

where \( \delta \) is the step size and \( \vec{v}_{q+1}^{\tau+1} \) is the unit-length motion vector of particle \( q \) at iteration \( \tau + 1 \). Since \( \vec{v}_{q+1}^{\tau+1} \) is provided by \( \Phi_q(\theta) \), it should be parallel to the flow. One way of computing \( \vec{v}_{q+1}^{\tau+1} \) is by taking the average ODF direction,

\[
\vec{v}_{q+1}^{\tau+1} = f \left( \frac{\int \Phi_q(\theta)d\theta}{\int \Phi_q(\theta)d\theta} \right)
\]

where \( f(\cdot) \) converts an angle \( \theta \) into a 2D vector of unit length. But as shown in Fig. 4 (a), in multimodal areas, Eq. (3) does not give a motion vector corresponding to an actual direction mode. Let us mention that this approach is similar to scene understanding methods relying on motion vectors averaged over time [1, 2, 15].

Alternatively, we sample the ODF in the area defined by the incoming direction \( \vec{v}_q^{\tau} \). As shown in Fig. 4 (b) and (c), given an opening angle \( \Delta \), sampling is done by the following function:

\[
\vec{v}_{q+1}^{\tau+1} = f \left( \frac{\int h_{\Delta}(\theta - \theta_i)\Phi_q(\theta)d\theta}{\int h_{\Delta}(\theta - \theta_i)\Phi_q(\theta)d\theta} \right)
\]

where \( \theta_i \) is the incoming direction vector angle and \( h_{\Delta}(\theta - \theta_i) \) is a kernel function. In our implementation, \( h_{\Delta}(\theta - \theta_i) \) is a 1D zero-mean Gaussian function with standard deviation set to \( \Delta/4 \) (a rectangular function of size \( \Delta \) could also be used). As shown in Fig. 4 (c), this allows particle to follow different directions depending on their incoming direction. This is a key advantage of our method which makes it robust to overlapping motion patterns.

In order to prevent a particle from being abruptly deviated by a noisy or corrupted ODF, Eq. (4) is incorporated within a temporal filter

\[
\vec{v}_{q+1}^{\tau+1} = \alpha \vec{v}_{q}^{\tau+1} + (1 - \alpha) f \left( \frac{\int h_{\Delta}(\theta - \theta_i)\Phi_q(\theta)d\theta}{\int h_{\Delta}(\theta - \theta_i)\Phi_q(\theta)d\theta} \right)
\]

where \( \alpha \in [0, 1] \) is a persistency constant (0.3 in our case). Algorithm 1 (in the supplementary material) summarizes the meta-tracking procedure for a particle \( q \).

**Stochastic sampling** The meta-tracking algorithm stops whenever one of the following criteria is satisfied: (1) if the iteration number has reached a MAX value (2000 in our case), (2) if the particle took a sharp turn between \( \tau \) and \( \tau + 1 \) (in our case, more than \( 20^\circ \)), and (3) if the particle enters an area where no activity has been recorded. Unfortunately, since ODFs are sometimes imprecise near the edges of occluding objects such as buildings and bridges, meta tracks next to these structures may be wrongly deviated towards areas with no activity. This results into unexpected stops and outlying meta tracks. As a solution, each
time a meta-track hits a region with no activity, the tracking procedure backs off \( l \) pixels and sends \( N \) new particles (usually \( l = 5 \) and \( N = 20 \)). These particles are tracked for \( 2l \) pixels following a stochastic ODF sampling. As opposed to deterministic sampling shown in Fig. 4, stochastic sampling randomly selects a direction with respect to the ODF shape. In this way, the \( N \) particles end up following different paths. If all those particles end up in an area where no activity has been recorded, the current meta-track is stopped. Otherwise, the valid tracks are averaged and Algorithm 1 restarts from that point.

### 3.4. Recovering Entry/Exit Areas and MPs

MPs are usually obtained by grouping tracks with similar shape, length and direction [22, 14]. Here, we propose a new method to group tracks without having to extract trajectory features or compute sophisticated inter-track distance metrics as is usually the case. Our method relies on a simple assumption: tracks within a single MP all start from a common entry point and finish into a common exit point. Our method thus aims at identifying the entry and exit areas and then cluster the meta-tracks accordingly.

Entry points are localized by analyzing the first point of each track\(^2\). Since the number of entry areas is not known \textit{a priori}, we use a bottom up hierarchical clustering method which does not assume a predefined number of clusters. In this way, each point starts in its own cluster and, iteratively, the nearest clusters are merged together as the algorithm moves up. The algorithm stops when the intra-cluster distance reaches a maximum value (between 20 and 50 in our case). Typical results are shown in Fig. 5. The small dots in the top images are the first and last points of each track while the large dots are cluster centers. MPs are obtained by grouping tracks connecting the same entry and exit areas.

These MPs are sorted according to their number of tracks, predominant MP having more tracks. Every MP with less than 5% of tracks are discarded.

### 4. Experimental Results

In order to gauge performances, we tested our method on 12 videos representing different challenges (see Fig. 8 and Fig.1 in the supplementary material). These videos contain between 300 and 8000 frames. They were taken from the UCF database, the changedetection.net [8] dataset, Youtube as well as our own database. Some videos have a small number of moving objects (e.g. MIT, Street Light), while others show highly crowded scenes (e.g. Marathon, Mecca). Some have a simple layout (e.g. Street Light, Boulevard) while others have a complex one (the 3 Roundabouts). Some have well structured dynamics (e.g. MIT, Rush Hour) while others show a fairly chaotic scene (e.g. Indian Market, Hanoi). The videos as well as the code can be downloaded from our project page.

![Figure 6. Our method performs well in areas where MPs overlap.](image)

The resulting meta-tracks for each video are shown in Fig. 8 and in the supplementary material. As can be seen, our meta-tracking method works well on every video, including the ones with complex layout and unstructured dynamics. It takes on average 30 secs to track 1000 particles on a 2.8 GHz laptop with Matlab code.

We compared our meta tracking results to those produced by three alternative solutions: two object-tracking methods and a vector field (VF) method similar to [1, 2, 10]. The object-tracking methods are Mean-shift [7] and the L1 tracker from Bao \textit{et al.} [4]. We used the OpenCV implementation for Mean-shift and Bao \textit{et al.}'s code for the L1 tracker. Since Bao \textit{et al.} did not specified how to initialize tracks, we manually initialized tracks by selecting every moving objects (or a subset of the moving objects for highly crowded scenes). As for the VF method, we implemented a tracker similar to [1, 2, 10] which uses a vector field to recover trajectories (or “sink paths”). For each method, small isolated tracklets have been filtered out.

\(^1\)An illustration of stochastic sampling can be downloaded from the following web page: \url{http://ilt.u-bourgogne.fr/benezeth/projects/GVSS2013/}.

\(^2\)Exit points are obtained by analyzing the last point of each track.
Table 1. Percentage of outlying tracks and average track length (ATL) obtained on 12 videos by 4 methods: our meta-tracking method, the Mean Shift object tracker [7], the L1 object tracker [4] and a vector field (VF) tracker. The “–” annotation are where a method has failed to recover any good tracks.

Table 1 shows for each method and each video, the percentage of outlying tracks (tracks that do not correspond to the actual flow or to any moving object) and the average track length (ATL). These results clearly show that object-tracking methods quickly diverge when the number of moving objects becomes large. This is especially true with Mean-Shift whose automatic initialization procedure is fairly sensitive. The ATL values show that inlying tracks produced by our method are longer than those of the object-tracking methods. As mentioned in [2], this is a strong indication that our method is more consistent and thus better suited to recover long-range MPs.

Table 2. Motion patterns segmentation results.

Our method and VF produce similar results on sequences with simple layout (Boulevard, Rush Hour, Street Light). However, our experiments reveal that in complex scenes, our method performs better than VF. As shown in Fig. 6, since our meta-tracker is based on ODFs, the bimodal nature of the flow in areas where MPs cross is correctly handled. This is clearly not possible with a vector field.

Fig. 7 shows the four most predominant MPs for sequences Escalator, Roundabout 1 and MIT. Our clustering Matlab code takes on average less than 1 second to process a scene. Quantitative results are presented in Table 2; ground truth for each video has been manually generated. The number of classes correspond to the dominant MPs.

Results in table 2 show the number of true and false detections for each sequence. False detections are usually generated in areas where motion is intricate. Let us emphasize good results obtained on complex sequences (Roundabout 1, Roundabout 2, Hanoi and Indian Market) for which we have no more than 1 false detection. Note that the size of the moving objects do not influence the output of the method.

### 4.1. Conclusion and Future Work

We introduced a method to recover motion patterns and entry/exit areas that works on sparse and crowded scenes, on structured and chaotic scenes and on scenes with simple and complex layout. This method is based on pixel-based ODFs which summarizes the flow of activity at each point of the scene. A grid of particles is laid on the scene and transported through the ODF field. This results into particle trajectories called meta-tracks. Those meta-tracks are then grouped together to form MPs. The experiments show that our method outperforms object-tracking methods, especially on dense scenes and produce better results than the vector field approach in scenes containing overlapping MPs.

### References

Figure 7. The four most predominant MPs extracted with our method for Escalator, Roundabout 1 and MIT.

Figure 8. Four results of our meta tracking method. From left to right: Marathon, Mecca, Indian Market, and Rush Hour.