# Urban Tracker: Multiple Object Tracking in Urban Mixed Traffic

Jean-Philippe Jodoin, Guillaume-Alexandre Bilodeau
LITIV lab., Dept. of computer & software eng.
École Polytechnique de Montréal
Montréal, QC, Canada
{jean-philippe.jodoin,gabilodeau}@polymtl.ca

Nicolas Saunier
Dept. of civil, geo. and mining eng.
École Polytechnique de Montréal
Montréal, QC, Canada
nicolas.saunier@polymtl.ca

## Abstract

*In this paper, we study the problem of detecting and tracking multiple objects of various types in outdoor urban traffic scenes. This problem is especially challenging due to the large variation of road user appearances. To handle that variation, our system uses background subtraction to detect moving objects. In order to build the object tracks, an object model is built and updated through time inside a state machine using feature points and spatial information. When an occlusion occurs between multiple objects, the positions of feature points at previous observations are used to estimate the positions and sizes of the individual occluded objects. Our Urban Tracker algorithm is validated on four outdoor urban videos involving mixed traffic that includes pedestrians, cars, large vehicles, etc. Our method compares favorably to a current state of the art feature-based tracker for urban traffic scenes on pedestrians and mixed traffic.*

## 1. Introduction

Detecting and tracking moving objects is one of the most important applications in computer vision. It is a vital tool in fields such as surveillance, security and transportation. In particular, video tracking is fostering new research in transportation engineering, for example in the study of the behavior and the safety of all road users, whether motorized or non-motorized, such as cars, cyclists and pedestrians [8]. New techniques are developed to perform road safety diagnosis based on the observation of road user interactions without having to wait for accidents to occur [16]. Video tracking allows acquiring more easily, at a lower cost and more accurately, larger amounts of data than could be done previously, typically manually. This leads to advances that can only be achieved by mining large amounts of observational data.

Considerable research has been done on tracking since it is a pillar of many video analysis techniques. Recent works have mostly focused on surveillance and crowd monitoring. These methods mostly rely on trained a pedestrian detector such as the one presented in [18] to localize the objects. Other works focus on the associations of multiple detections like [4]. This type of methods works well to track multiple pedestrians in complex scenes. Unfortunately, this is not suited for urban tracking. Urban tracking has its own challenges that cannot be easily addressed by these methods due to the large variety of road users such as pedestrians, cyclists, cars, trucks, etc. All these objects have different shapes and appearances. If pedestrians have similar shapes, their appearance, in particular the garment colors, varies widely and they are non-rigid. Vehicles have different colors and shapes, in particular when their pose changes as they move through out the scene and they turn at intersections. Tracking all possible road users in outdoor urban traffic scenes would therefore require a large number of detectors for the various types of objects and their main poses. Although a multi-view approach to car detection has been proposed by [14], it is difficult to generalize to all kinds of vehicles and road users. Tracking by detection in traffic scenes is only practical if the variety of poses and road user types is limited, e.g. for highways. Because of this, only the motion information can be used to detect the objects. This task can be achieved using either optical flow or background subtraction. In both cases, the resulting detections are either fragmented or merged which must be dealt with by the tracking algorithm. Recent methods based on object detectors cannot address these problems as they assume that only extra or missing detections are possible.

The objective of this work is to design a tracker able to handle multiple objects of various shapes and sizes. Since we want to collect information for road safety at intersections, we focus on outdoor urban traffic scenes captured using a single camera. We propose a tracking approach with the flexibility to detect and track multiple types of moving objects without prior knowledge of those objects. It uses moving object detection (background subtraction) that allows handling multiple objects while being able to

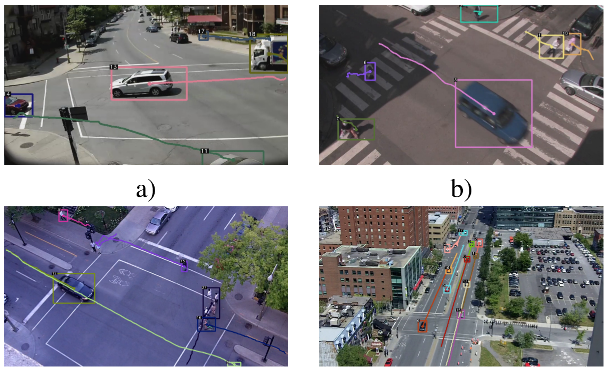
a) b) c) d)

Figure 1. Sample results of Urban Tracker on scenes of interest. a) Sherbrooke, b) Rouen, c) St-Marc and d) Rene-Levesque

detect and track any unforeseen type of road users. However, background subtraction complicates data association because objects may occlude each other or become fragmented. To handle fragmentation and occlusion, we propose an object model that is based on feature points and spatial information computed on the detected area. Data association is performed using this model for each detected object. The object can be thus followed even in the presence of partial occlusions and recovered after a short total occlusion. We named our tracker: Urban Tracker. Some sample images of processed scenes with results are shown in figure 1.

## 2. Related Work

Road user tracking is a popular application for tracking algorithms. There are two main types of applications: highway traffic and urban traffic scenes. Tracking vehicles on highways is easier than in urban areas as there are fewer types of objects (only motorized vehicles of various sizes), little change in the orientation of the vehicles and few known entry and exit points. Cameras are also usually located much higher than in urban scenes, which reduces the occurrence of occlusions. Tracking vehicles on highways is more challenging when the traffic is slower because the inter-vehicle space is significantly reduced, increasing the occlusion between vehicles. In outdoor urban areas, traffic includes also pedestrians and cyclists (and even wheelchairs), and more complicated trajectories, with vehicles turning at intersections, stopping and parking, and many more entry and exit points in the scene. Different computer vision methods have thus been developed for these two types of applications.

Coifman *et al.* [5] use the Kanade-Lucas-Tomasi tracker [17] to find good feature points and track them. The feature tracks are then grouped based on common motion constraints by computing the difference between the minimal and the maximal distance between two tracks. This works well for a highway even in the presence of partial occlusions since the common motion constraint allows distinguishing moving cars as long as they move at sufficiently different speeds. Jun *et al.* [9] used background subtraction to estimate the vehicle size. They used a watershed segmentation technique to over-segment the vehicle. The over-segmented patches are then merged using the common motion information of tracked features points. This allows to segment vehicles correctly even in the presence of partial occlusion.

Saunier *et al.* [15] adapted the work by Coifman *et al.* [5] to track all types of road users in outdoor urban intersections, by detecting continuously new features and adding them to current feature groups. The challenge is to find the right parameters to segment objects moving at similar velocities, while at the same time not over-segmenting smaller non-rigid objects such as pedestrians. Another tracking algorithm for urban traffic scenes is described by Kim *et al.* [10]. This method combines background subtraction and feature tracking approaches with a multi-level clustering algorithm based on Expectation-Maximization (EM) to handle the various object sizes in the scene. The resulting algorithm tracks various road users such as pedestrians, vehicles and bicycles online and the results can then be manually corrected in a graphical interface.

Our online tracking algorithm, Urban Tracker, also combines feature tracking and background subtraction. In contrast to the work in [10], which uses Expectation Maximization to cluster feature trajectories in objects, we rely instead on background subtraction for the same task. We thus consider features as groups inside common regions instead of groups with common motion. This allows us to discriminate between vehicles having common trajectory and speed. A state machine and interframe blob association are used to handle the occlusion and segmentation problems of the background subtraction. Finally, this feature grouping procedure allows us to achieve better tracking results on non-rigid object than methods relying on common motion criteria like [15].

## 3. Methodology

Our tracking algorithm is a combination of blob and feature tracking. Blobs are used for size estimation, feature grouping and data association, while features are used for data association and occlusion resolution. The high-level diagram of our algorithm is shown in figure 2. The method consists in three main steps applied on each frame. The first step is blob extraction where the foreground blobs are extracted in the video frame and blob models are calculated. The second step is the blob tracking. It involves the tracking of the individual blobs from one frame to the next. The last step is the object tracking. It is based on the notion of object tracks and track states. An object track is a sequence

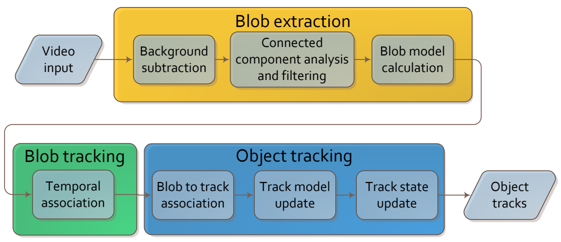

Figure 2. High-level view of the system

of temporal detections of a real world object. For the rest of this paper, we will be referring to the term object tracks as tracks. One or many blobs at each frame may represent a track. The state of an object track is used for tracking and represents its life cycle in the scene (entering, exiting, lost, etc.). This step uses the blob association information to resolve the ambiguity between the blobs and the object tracks: it is the step of the algorithm that is responsible for resolving occlusion, blob fragmentation and for updating the object model. In this section, the three steps of the algorithm will be presented in more details.

## 3.1. Blob extraction

The first step of blob extraction consists in applying a background subtraction algorithm to the input video frames. To decrease noise, we use a Gaussian blur with a 5x5 kernel before doing the background subtraction. We then apply the ViBe background subtraction method [1]. This method was selected because it is fast and it is among the top performers on the original *changedetection.net* benchmark [7]. Once the foreground pixels are detected, we compute the spatially connected pixels to get blobs. This gives us a mask of the moving blobs in the scene. We then slightly dilate this mask to fill the holes. Blob smaller than $T_m$ pixels are filtered out. We have slightly modified the ViBe method to better handle intermittent object motion that occurs often in urban scenes since vehicles will park or stop on red light. At each frame, the blobs are analyzed to verify if the internal pixel intensity changed enough (to assess if the blob is moving or not). This verification is done by calculating the absolute pixel intensity difference between the current frame and the previous one in each blob. We then count the number of pixels with an intensity change ($np_c$) over a threshold of $\theta$ (we have used $\theta = 4$ in order to accept pixel variation due to noise in our video). If too few pixels have changed ($np_c/area(blob) < 0.1$), the ViBe background model is updated with the pixels of the current frame for all pixels located inside the blob in order to remove the "ghost blob". The last step of blob extraction is to compute the model of each blob for the next step of interframe blob association as described in section 3.2.

The blob model is composed of the blob size, the blob position and the feature points located inside it. The BRISK feature point detector is used with 3 octaves and a detection threshold of 10 [11]. This number of octaves is the default value in OpenCV [3]. A detection threshold of 10 is used instead of the default value of 30 because the latter did not yield enough points on the objects to correctly resolve the occlusions. However, this lower threshold value tends to create a higher number of false matches because the points are less discriminative. The feature point descriptor is the FREAK binary descriptor [13] since it shows superior performance to the BRISK descriptor. This descriptor presents some invariance to rotation and scale while being fast to calculate due to its binary nature. It was used with 3 octaves like BRISK and a scaling pattern of 22 px. This gives us a good robustness to scale changes, which is useful when an object reappears after an occlusion. The disadvantage of those parameters is that we have a featureless border of 44 px (as represented in figure 3) due to FREAK's scaling pattern size.

The track model is composed of a sequence of blob models. Unmatched keypoints from the concatened blob models are removed in order to keep only the truly useful keypoints.

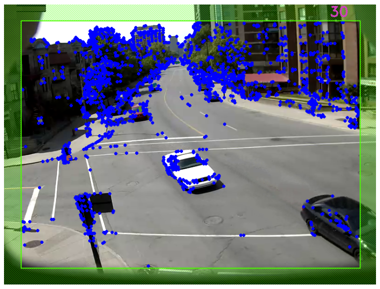

Figure 3. Feature positions when using the FREAK descriptor with a 22 px scaling pattern

## 3.2. Blob tracking

Blob tracking can be considered as low-level tracking since it deals with the temporal association between blobs in two consecutive frames. The segmentation issues and occlusion problems are dealt with in the object track building part described in section 3.3. We are interested in the following blob tracking events: blobs entering the scene, blobs exiting the scene, lost blobs, fusion between multiple blobs and the splitting of a blob into multiple smaller blobs. The blob model defined in the previous section is used to detect these events. The first step is to calculate the correspondence between the descriptors of the feature points located inside the set of blobs $B_{t-1}$ in the previous frame and the set of blobs $B_t$ in the current frame using the Hamming distance. The tests defined in [12], the ratio and the symmetry test, are run for each pair of points to filter bad matches.

The ratio test computes the distance ratio between the best match over the second best match in order to filter out less distinctive point matches. The symmetry test consists in verifying if the matching is mutual for the two points in the pair. Finally, to avoid bad matches, associations are considered valid only if matching blobs have a minimum of four feature matches.

This results in $1 \rightarrow 1$, $1 \rightarrow N$, $N \rightarrow 1$, $0 \rightarrow 1$ and $1 \rightarrow 0$ associations on a per blob basis with $N > 1$. Since we have a significant featureless border and because smaller objects will have fewer features, we also measure blob overlap for the blobs that do not possess enough feature points. This is only applied to blobs that are unmatched using features since features are more reliable for association, especially when multiple objects are near each other. The dimension and position of the blob model are used to calculate the area of overlap between the unmatched blobs $B^u_{t-1}$ and $B^u_t$ and the matched ones $B^m_{t-1}$ and $B^m_t$. If a blob overlaps with many other blobs, we keep the one with the biggest area of overlap and add the overlap association to the associations obtained using feature matches. The remaining blobs with no overlap $B^u_{t-1}$ are considered lost and $B^u_t$ are considered as new blobs. All $1 \rightarrow N$ associations are then considered a split, $1 \rightarrow 1$ a direct blob track and $N \rightarrow 1$ a merge. There are no $N \rightarrow M$ associations by construction.

## 3.3. Track building

The track building part of the algorithm is responsible for the association between the blobs and the object tracks. This allows coping with the difficulties of real data coming from an uncontrolled environment. It handles oversegmentation and undersegmentation of blobs and manages the lifecycle of object tracks, using the state machine shown in figure 4. The transitions are determined by the blob associations.

### 3.3.1 Track states

A track stays in the normal state as long as it is composed of $1 \rightarrow 1$ blob associations (transition $a_6$). The track model is updated and the state remains unchanged. If the blob associated to an object track in the normal state disappears (transition $a_5$, $1 \rightarrow 0$ association), the object track state is changed to lost. If the object track remains lost for more than a given number of frames ($N_r$), this track is deleted from the list of tracked objects (transition $a_{12}$). When a new unassociated blob is detected, two actions are possible:

1. If the blob model is similar (contains 4 pairs of keypoint matches) to a track in the lost state, the track state becomes normal again (transition $a_{11}$). 2. If it cannot be linked to a lost track, a new object track is created in the hypothesis state (transition $a_0$, $0 \rightarrow 1$ association). If no association can be found in the next 3 frames, it will be

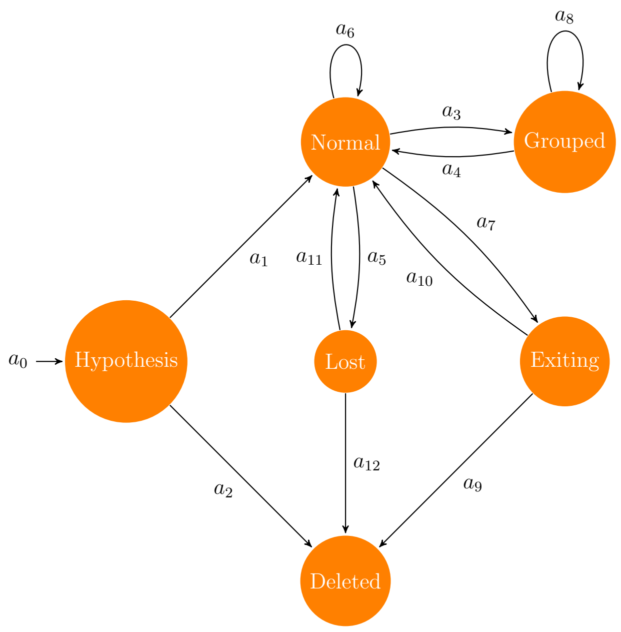

Figure 4. Object state machine

deleted directly (transition $a_2$, $1 \rightarrow 0$ association). This allows us to remove noise and unstable tracks. If the object is successfully tracked for 3 frames, the state of the track will be changed to normal (transition $a_1$, $1 \rightarrow 1$ association).

If multiple blobs associated to existing tracks are occluding each other (transition $a_3$, association $N \rightarrow 1$), we create a group of tracks with all the track models, and each object track will be updated separately in the group (more details at section 3.3.2). If the group of tracks splits (association $1 \rightarrow N$), the tracks that are split from the group may go in the normal state (transition $a_4$) or remain inside the group (transition $a_8$). If the association $1 \rightarrow N$ is linked to a track in the normal state, this might be a case of initial undersegmentation (two object tracks entering in the same blob in the scene) or a case of fragmentation. The split track will be separated and created in the normal state (transition $a_6$). If a track touches the border of the scene, it changes to the exiting state (transition $a_7$). If it is not tracked in the next frame, it goes in the deleted state (transition $a_9$). If the track changes direction and stops touching the border, a verification of its model is done to verify that it is still the same track (we require three feature matches for the model to be considered the same). If it is the same track, the track state changes back to normal (transition $a_{10}$). If it is a different track, the track is deleted and a new track is created. The exiting state is needed since the associations found on the border of the images are calculated only using the blob shape (due to the featureless border). Since exiting tracks often cross entering tracks, we need to do further verification to make sure they are still the same object.

### 3.3.2 Undersegmentation handling

A blob is considered undersegmented if it is associated to multiple tracks. This situation can occur when multiple objects are occluding each other or because of the misclassification of background pixels as foreground. If multiple tracks enter the scene in the same blob and they split later while they are visible, we will create new object tracks (transitions $a_6, a_8$), and we will use the model of each blob to retrieve the movement history of the tracks before they split. On the other hand, if multiple distinct tracks merge inside a single blob (transition $a_3$), we will be able to follow them individually by using the model that was learned before they merged. The feature matches between the feature points of each object track model and the new merged blob are computed: matching points of the new blob will be added to a list of previously matching features and the track model is updated with them. In order to evaluate the position of the bounding box of individual objects, we will use the matching features. This process is illustrated on figure 5.

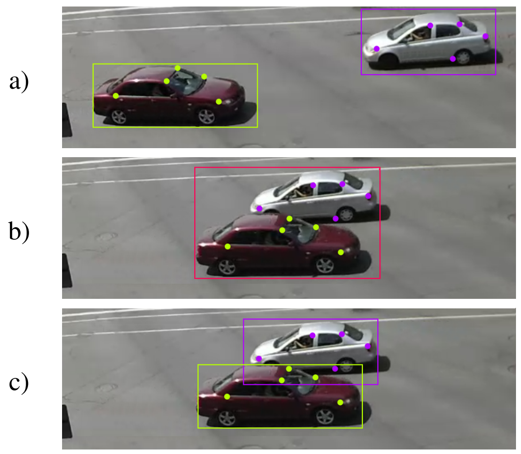

Figure 5. Blob estimation process: a) Previous observation bounding boxes with points b) Undersegmented blob with current points c) Estimated bounding boxes

Figure 5a) represents the previous observation with feature points and known bounding box. In figure 5b) we can see the undersegmented blob with the same features. For all of theses features, we calculate their position relative to the centroid of the bounding box in a). This relative position is added to their current position in b) in order to get an estimate of the centroid position. We repeat the process for all features and get a list of estimated positions of the centroid. In order to reduce noise, we select the median values on the x and y axis to estimate the box centroid. The dimension of the bounding box is also the median size of previous observations (before the object became part of the group of object in the undersegmented blob). The final bounding boxes of each track can be seen on figure 5c).

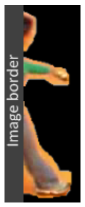

Figure 6. Oversegmentation when a pedestrian enters the scene

### 3.3.3 Oversegmentation handling

An oversegmented track happens when multiple blobs represent the same track. Oversegmentation of object tracks can occur due to: 1) errors in the background subtraction, 2) concave objects entering the scene (as shown in figure 6) or 3) static objects occluding in the scene. To resolve cases 1) and 2), we use the hypothesis state as a temporal buffer before creating permanent tracks (transition $a_1$). This gives time to the background subtraction to stabilize and since tracks in the hypothesis state are automatically merged with spatially close tracks, the track is less likely to be split into several parts. To resolve 3), we use spatial information and the result of the blob association. Before splitting a track, we verify that the blobs diverge enough spatially by dilating them by a factor $D_b$ (10 % in our case). If there is an overlap between the blobs, the split is delayed since the object tracks are still too close to conclude that they are two separate tracks. The use of the bounding boxes and a dilatation factor makes this test more robust, which is crucial in urban scenes where there is considerable variation in object size.

### 3.3.4 Exiting tracks

Exiting tracks are problematic because their blob may merge with entering tracks (as seen in figure 7). The featureless border of the image complicates this issue even further because the quality of blob association is more error prone in this area, as it cannot rely on the keypoints. Since this case is quite common in urban scenes, the exiting state is introduced. As soon as a track touches the border of the image, its state changes to exiting. From that point on, three situations may occur: 1) the track leaves the scene, 2) the track does not leave the scene and starts going back toward the middle of the scene, 3) the track leaves the scene and its blob is merged with an entering track. Case 1 requires no particular processing. The critical issue is to distinguish case 2 from 3. If the track leaves the border and goes toward the middle, its identity is verified using feature points. Objects in this situation need to have at least three feature points to keep their current identity (case 2), or else we assume it is case 3 and the object track is given a new id. In case 3, the history of the new track is recovered from the previous track model. In order to estimate the frame that

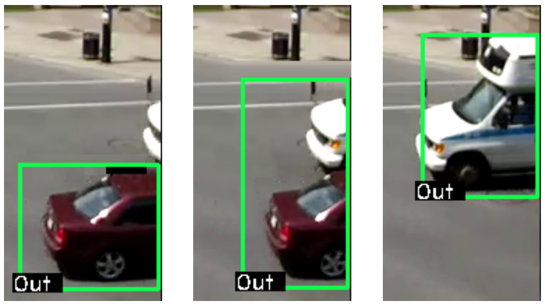

Figure 7. Id change problem when an object exits and another enters at the same time. These are cropped images of the right side of the scene. a) The red car leaves the scene. b) The white bus is part of the same blob as the red car. c) The white bus takes the id of the red car.

the blob became more of the new track and less of the exiting one, we look at the track blobs when they were in the exiting state. The frame we are looking for is the one when the object blob was the smallest since it indicates the instant when the blob size started to be modified by the entering track. This allows us to separate the blob of the entering track from the exiting track.

## 4. Results

Urban Tracker was tested on four video sequences and compared with another tracker called Traffic Intelligence (TI) [15][1].

### 4.1. Evaluation Methodology

The four outdoor test sequences are shown in figure 1. The Sherbrooke video is a 1001-frame 800x600 sequence with cars, trucks and pedestrians moving at an intersection. There are 5 pedestrians and 15 cars in the annotated video. The traffic is light but it still involves many occlusions between the road users. The Rouen video sequence is a 600-frame video with 11 pedestrians, 4 cars and 1 bicycle. Multiple occlusions occur between pedestrians. The camera angle is quite different from the Sherbrooke video. The St-Marc sequence is a 1000-frame 1280x720 video of an intersection that contains 7 cars, 2 bicycles and 19 pedestrians. Interactions are similar to those of Rouen. Finally, the Rene-Levesque sequence is a 1000-frame 1280x720 video from a very high camera covering three intersections. The farthest intersections and pedestrians were not annotated in this video because the objects are very difficult to distinguish even by a human. The annotated part of the scene contains 29 cars and 2 bicycles.

For all videos, each object is annotated as soon as it starts moving until it leaves the scene. An object leaving the scene and reentering later is considered a different object. All objects were annotated (id, center position, and bounding box corners) even if they were only partially visible. Also, a mask is applied on the Sherbrooke and Rene-Levesque video to disable tracking outside the region of interest closest to the camera.

The CLEAR MOT metrics are used to quantify the quality of the tracking [2]. They are composed of two main metrics, the multiple object tracking precision (MOTP) and the multiple object tracking accuracy (MOTA). The MOTP is independent of tracking errors such as id changes and it measures the average precision of the instantaneous object matches, which can be estimated in two different ways. For the evaluation of a tracker yielding a bounding box such as Urban Tracker, the average of the bounding box overlaps for all matching objects can be computed. For a tracker yielding only a centroid position such as TI, the precision of the tracker is the average distance between the centroid of the ground truth bounding box and the tracker output centroid. To compare with TI, we use the second definition. The MOTA is a metric that takes into account the number of misses, the number of false positives and id changes. To calculate the metrics, a threshold must be set to evaluate the maximum distance an object can be from the ground truth object to be considered a valid match. For each video, the threshold was chosen to impose a minimal overlap of 50 %, like in the PASCAL VOC challenge [6], which translates into a maximum diagonal distance of 90 px for Sherbrooke, 164 px for Rouen, 113 px for St-Marc and 24 px for Rene-Levesque. Smaller MOTP values and larger MOTA values reflect better performance.

### 4.2. Experimental Results

To generate the experimental results, we have selected some parameters for each video for both algorithm. For Urban Tracker (UT), the parameters that were changed are the number of frames to look for a lost object ($N_r$), the fragmentation factor ($D_b$) and the minimum blob size ($T_m$). The parameters used are reported in table 1. Our results were obtained without any form of camera calibration. To be as fair as possible, for TI, we have asked the authors to provide us with good parameters for the test videos. An homography was used with TI when available. The parameters that were adjusted were the connection distance ($d_{con}$), the segmentation distance ($d_{seg}$) and the number of features ($n_f$). The parameters for each dataset are reported in table 2.

The results of each video are presented in table 3. We observe that the MOTA for UT are higher than TI. This can be explained by the use of background subtraction which allows us to detect objects of multiple types and sizes. The MOTA of objects can vary by type because of the more

[1]Available under an open source license at https://bitbucket.org/Nicolas/trafficintelligence (revision 8f8f437 September 3th, 2013)

| Video | Parameters | Homography |
|---|---|---|
| Sherbrooke | $N_r$ = 1160, $D_b$ = 0.1, $T_m$ = 300 | no |
| Rouen | $N_r$ = 150, $D_b$ = 0.7, $T_m$ = 380 | no |
| St-Marc | $N_r$ = 1160, $D_b$ = 0.1, $T_m$ = 300 | no |
| Rene-Levesque | $N_r$ = 900, $D_b$ = 0.2, $T_m$ = 50 | no |

Table 1. Parameters used in UT with each video.

| Video | Parameters | Homography |
|---|---|---|
| Sherbrooke | $d_{con}$ = 4 m, $d_{seg}$ = 1.7 m, $n_f$=1000 | yes |
| Rouen | $d_{con}$ = 25 px, $d_{seg}$ = 25 px, $n_f$=1000 | no |
| St-Marc | $d_{con}$ = 10 px, $d_{seg}$ = 40 px, $n_f$=1000 | no |
| Rene-Levesque | $d_{con}$ = 10 px, $d_{seg}$ = 40 px, $n_f$=2000 | no |

Table 2. Parameters used for TI with each video.

complex interactions between the objects and the resulting occlusions. Further analysis shows that the MOTA of TI vary widely between the types of objects. This can be explained by the reliance on a grouping threshold that does not handle well non-rigid objects and objects of varying sizes. In these videos, parameters for TI were optimized for the more frequent type of objects in the scene.

For the Sherbrooke video, UT groups two pedestrians as one for the entire duration of the video since they stay closely together. If we consider both pedestrians as one in the ground truth, the MOTA for pedestrians increase from 0.6809 to 0.8764, which is very similar to the score for cars. For Rouen and St-Marc, most of UT tracking errors are due to grouping problems between pedestrians. UT has a bit more difficulty to track cars in Rene-Levesque then in the other videos. This is because we start tracking cars very far from the camera while they occlude each other and it takes some time for the algorithm to separate them. These observations further confirm that the discrepancy between the MOTA of pedestrians and cars for UT is caused by occlusions between them and not because of their different characteristics, appearnace, etc.

UT scores for bicycles are very high for Rouen and St-Marc, and it is low for Rene-Levesque. This can be explained by the fact that since the camera is very high, one of the two bicycles in Rene-Levesque is very small and difficult to detect (even for a human). So what we actually have is one bicycle tracked correctly for all the sequence while the other one remains undetected for almost all the sequence. The same behavior can be observed with TI for the Rene-Levesque video.

UT's precision is higher than TI for most of the video and type of object. Cases where the precision of UT is lower are for the cars in the Sherbrooke video and the bicycles in the Rouen video. For the cars in Sherbrooke, this can be explained by some bad estimation on the image boundaries and the presence of shadows. For the bicycles in Rouen, the slightly higher MOTP for TI can be explained by the shadows deforming the blobs. For TI, the object types with a low accuracy in tracking also suffer from a very low precision. This is a side effect of using threshold optimized for specific object types. For instance, when we optimised the threshold for pedestrians, only parts of the vehicles are then tracked, which moves the centroid to the center of that part.

| | | Method | | | |
|---|---|---|---|---|---|
| | | UT | | TI [15] | |
| Video | Type | MOTA | MOTP | MOTA | MOTP |
| Sherb. | Cars | **0.8928** | 10.53 px | 0.8253 | **7.42 px** |
| | Pedestrians | **0.6809** | **6.64 px** | 0.0141 | 11.98 px |
| | All objects | **0.7771** | 8.66 px | 0.3841 | **7.54 px** |
| Rouen | Cars | **0.8965** | **9.73 px** | 0.1853 | 66.69 px |
| | Pedestrians | **0.8050** | **13.64 px** | 0.6467 | 20.04 px |
| | Bicycles | **0.9270** | 14.13 px | 0.8686 | **13.11 px** |
| | All objects | **0.8234** | **13.09 px** | 0.5885 | 24.20 px |
| St-Marc | Cars | **0.8887** | **10.9 px** | -0.1778 | 38.99 px |
| | Pedestrians | **0.7216** | **4.99 px** | 0.6926 | 10.44 px |
| | Bicycles | **0.9895** | **6.70 px** | 0.8952 | 7.46 px |
| | All objects | **0.7567** | **5.97 px** | 0.6018 | 14.58 px |
| Rene-L. | Cars | **0.8038** | **3.02 px** | 0.5474 | 5.23 px |
| | Bicycles | **0.2509** | **2.18 px** | 0.2321 | 3.14 px |
| | All objects | **0.7267** | **2.97 px** | 0.5029 | 5.10 px |

Table 3. CLEAR Metrics for the videos. All results are shown separated by type and together. Boldface indicates best results. Sherb refers to the Sherbrooke video and Rene-L. refers to the Rene-Levesque video.

In order to evaluate the sensibility to parameters, both algorithms were run on each video using the parameters used previously in the other videos. These results are reported in table 4. Results were not reported for TI on the Sherbrooke video since the parameters were taking into account a homography and this homography is not applicable to other scenes.

As we can see, parameter choice has less impact on the results of UT than TI. The most significant change in accuracy comes from the use of Rene-Levesque's parameters on the Rouen video. This can be explained by the high fragmentation in the Rouen video that is not taken into account in the parameters of the Rene-Levesque video. The precision of UT is only slightly affected by the change of parameters. For TI, we can see the importance of the choice of parameters. When using the St-Marc and Rene-Levesque parameters on the Sherbrooke video, the accuracy drops significantly. Although it is has been shown that TI is less sensitive to parameters when using homographies [15], TI needs different grouping and segmentation parameters to deal with objects of different sizes: without classification and different sets of parameters for different object types, its accuracy is therefore significantly affected in scenes with objects of different sizes, leading to overgrouping and undergrouping of objects.

## 5. Conclusion

This paper presented Urban Tracker, a new tracking algorithm based on recent binary descriptors and background subtraction technique. The combination of both methods

| Sherbrooke parameters | | | | |
|---|---|---|---|---|
| | UT | | TI [15] | |
| Video | MOTA | MOTP | MOTA | MOTP |
| Rouen | 0.7697 ⇓ **5.37%** | 14.28 px ⇑ **1.19 px** | N/A | N/A |
| St-Marc | 0.7567 ≈ **0.00%** | 5.97 px ≈ **0.00 px** | N/A | N/A |
| Rene-L. | 0.7261 ⇓ **0.06%** | 2.80 px ⇓ **0.17 px** | N/A | N/A |

| Rouen parameters | | | | |
|---|---|---|---|---|
| | UT | | TI [15] | |
| Video | MOTA | MOTP | MOTA | MOTP |
| Sherb. | 0.7807 ⇑ **0.36%** | 8.69 px ⇑ **0.03 px** | 0.3794 ⇓ 0.47% | 13.80 px ⇑ 6.26 px |
| St-Marc | 0.6952 ⇓ **2.03%** | 7.07 px ⇑ **0.72 px** | 0.4219 ⇓ 18.0% | 20.93 px ⇑ 6.35 px |
| Rene-L. | 0.6741 ⇓ **5.26%** | 3.04 px ⇑ **0.07 px** | 0.3055 ⇓ 19.7% | 5.77 px ⇑ 0.67 px |

| St-Marc parameters | | | | |
|---|---|---|---|---|
| | UT | | TI [15] | |
| Video | MOTA | MOTP | MOTA | MOTP |
| Sherb. | 0.7771 ≈ **0.00%** | 8.66 px ≈ **0.00 px** | -0.8599 ⇓ 124.40% | 19.90 px ⇑ 12.36 px |
| Rouen | 0.7697 ⇓ **5.37%** | 14.28 px ⇑ **1.19 px** | 0.4427 ⇓ 14.58% | 30.64 px ⇑ 6.44 px |
| Rene-L. | 0.7261 ⇓ **0.06%** | 2.80 px ⇓ **0.17 px** | 0.2056 ⇓ 29.73% | 5.95 px ⇑ 0.85 px |

| Rene-Levesque parameters | | | | |
|---|---|---|---|---|
| | UT | | TI [15] | |
| Video | MOTA | MOTP | MOTA | MOTP |
| Sherb. | 0.7787 ⇑ **0.16%** | 9.25 px ⇑ **0.59 px** | -0.5029 ⇓ 88.70% | 17.29 px ⇑ 9.75 px |
| Rouen | 0.7122 ⇓ 11.12% | 15.59 px ⇑ **2.50 px** | 0.5173 ⇓ **7.12%** | 34.17 px ⇑ 9.97 px |
| St-Marc | 0.6781 ⇓ **7.86%** | 7.74 px ⇑ **1.77 px** | 0.3352 ⇓ 26.66% | 16.78 px ⇑ 2.20 px |

Table 4. Results obtained by using the same parameters for all videos. For each column, the number on the left of the arrow represents the results and the number on the right represents the difference between the MOTA and the MOTP for the optimized set of parameters and the scene specific parameters. The MOTA difference is express in percent points. For the MOTA column, a ⇓ represents a lower tracking accuracy while for the MOTP column, a ⇑ represents a lower tracking precision.

allows us to track road users independently of their size and type. Experimental results show that it performs better than a current state of the art tracker for urban traffic scenes and pedestrians on four real urban traffic videos. Our main contribution is a new tracker designed specifically for urban tracking that requires no prior knowledge (camera calibration) while needing very few intuitive parameter adjustments. To improve object localization in case of partial occlusion, we also proposed a bounding box estimation method. Results show balanced performance for the tracking of pedestrians, bicycles and vehicles. Finally, we also propose four new public urban videos with annotations and a metrics evaluation tool in order to inspire other to work on urban tracking since it is still an open problem in the general case. These videos and tools are available on: http://www.jpjodoin.com/urbantracker/.

## 6. Acknowledgments

This work was supported by scholarships from FRQ-NT and NSERC, and partially supported by NSERC discovery grant No. 311869-2010 and 402320-2011.

## References

[1] O. Barnich and M. Van Droogenbroeck. ViBe: A universal background subtraction algorithm for video sequences. *IEEE Transactions on Image Processing*, 20(6):1709–1724, June 2011. 3

[2] K. Bernardin and R. Stiefelhagen. Evaluating multiple object tracking performance: the CLEAR MOT metrics. *J. Image Video Process.*, 2008:1:1–1:10, jan 2008. 6

[3] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000. 3

[4] A. A. Butt and R. T. Collins. Multi-target tracking by lagrangian relaxation to min-cost network flow. In *CVPR 2013*, pages 1846–1853, June 2013. 1

[5] B. Coifman, D. Beymer, P. McLauchlan, and J. Malik. *A real-time computer vision system for vehicle tracking and traffic surveillance*, volume 6. 1998. 2

[6] M. Everingham, L. Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *Int. J. Comput. Vision*, 88(2):303–338, June 2010. 6

[7] N. Goyette, P. Jodoin, F. Porikli, J. Konrad, and P. Ishwar. Changedetection.net: A new change detection benchmark dataset. In *CVPR 2012 Workshops*, pages 1 –8, June 2012. 3

[8] K. Ismail, T. Sayed, N. Saunier, and C. Lim. Automated analysis of pedestrian-vehicle conflicts using video data. *Transportation Research Record*, 2140:44–54, 2009. presented at TRB 2009. 1

[9] G. Jun, J. K. Aggarwal, and M. Gokmen. Tracking and segmentation of highway vehicles in cluttered and crowded scenes. WACV 2008, page 1–6, Washington, DC, USA, 2008. IEEE Computer Society. 2

[10] Z. Kim. Real time object tracking based on dynamic feature grouping with background subtraction. In *CVPR 2008*, pages 1–8, 2008. 2

[11] S. Leutenegger, M. Chli, and R. Siegwart. BRISK: binary robust invariant scalable keypoints. In *ICCV 2011*, pages 2548–2555, 2011. 3

[12] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, Nov. 2004. 3

[13] R. Ortiz. Freak: Fast retina keypoint. In *CVPR 2012*, pages 510–517, Washington, DC, USA, 2012. IEEE Computer Society. 3

[14] M. Ozuysal, V. Lepetit, and P.Fua. Pose estimation for category specific multiview object localization. In *CVPR 2009*, Miami, FL, June 2009. 1

[15] N. Saunier and T. Sayed. A feature-based tracking algorithm for vehicles in intersections. In *The 3rd Canadian Conference on Computer and Robot Vision, 2006*, pages 59–59, 2006. 2, 6, 7, 8

[16] N. Saunier, T. Sayed, and K. Ismail. Large scale automated analysis of vehicle interactions and collisions. *Transportation Research Record*, 2147:42–50, 2010. presented at TRB 2010. 1

[17] J. Shi and C. Tomasi. Good features to track. In *CVPR 1994*, pages 593–600, 1994. 2

[18] J. Yan, X. Zhang, Z. Lei, S. Liao, and S. Z. Li. Robust multi-resolution pedestrian detection in traffic scenes. In *CVPR 2013*, pages 3033–3040, Washington, DC, USA, 2013. IEEE Computer Society. 1