A Bigraph Based Multi-Object Tracking System

JiaXin Wang
University of the Chinese Academy of Sciences
Beijing, China
Email: wangjiaxin15@mails.ucas.ac.cn

MaCui Xia
The Chinese Academy of Sciences
Beijing, China
Email: cuixia@iscas.ac.cn

TengDong Xing
The Chinese Academy of Sciences
Beijing, China
Email: dongxing@iscas.ac.cn

***Abstract*—Multi-Object Tracking is an important task in Computer Vision, which has gained increasing attention from academia to industry. In this paper we propose a bigraph based model, which use motion detection algorithm to obtain foreground mask, then we do data association novelly using Hungarian Algorithm and Local Invariant Feature Matching. To overcome the occlusion and oversegmentation in objects, we proposal a status machine which can totolerate various detection error in previous steps. Our tracking method can tracking multiple objects simultaneously, no matter the objects are car, bike, people and under various environment. The experiment show our algorithm outperforms the state-of-art real-time multi-object tracking algorithms.**

I. Introduction

Videos in Surveillance system has become the biggest data today [**?**]and the number of Fixed Videos are increasing every day. To use the information hidden in those mass data, we combine Motion Detection and Multi-Object Tracking technology to implement real-time object tracking system. We using raw video from camera as input, and the output is the trajectory of each objects and their relationship (split and merge) with each other. The motion detection algorithm convert the raw video into binary mask video, which contain errors as small size false positive due to background noise, medium size false negative due to background cluster, large size false positive/negative due to dynamic background or intermittent object motion. What's more, when the light change dramatically or in some extreme weather, the motion detection algorithm may crash. So our tracking algorithm need to be robust enough to solve errors from motion detection and the split, merge, born and miss challenging from occlusion and limit field of view. For the consideration of real-time and generalization, we prefer to use binary mask video as input for multi-object tracking algorithm which convert binary mask video into trajectory and object's relationship, rather than use feature-based detection result as input for multi-object tracking algorithm[**?**].

The limits for feature-based detection algorithm are obviously. Firstly, we need to build database to train the algorithm and for different objects we need build different database and train different detectors. For tracking a car, we need a car detector, for tracking a people, we need a people detector. Secondly, the feature-based detection algorithm often loss some objects and the detection results depend on the size of object. Usually, the feature based detection algorithm based on slide windows, so the object not fit in the slide window size cannot be detected by feature based detection algorithm well. Thirdly, for the objects have partly occlusion with background or other objects, the feature-based detection algorithm may also failed, that's because the detector often trained by objects without occlusion, and the detector lack the ability to detect objects with partial occlusion. Finally, feature-based detection algorithm is far slower than motion detection algorithm, the motion detection algorithm is more suitable for real-time application.

Challenging remains in motion detection, feature point matching and multi-object tracking. Here we focus on how to create a robust multi-object tracking system based on current state-of-art motion detection algorithm. For most surveillance video, we can simply use motion detection algorithm to obtain objects. Usually, motion from those raw video is caused by object (human or vehicle). However, we need notice that motion from scene can also be caused by tree in wind. So for 2D fixed video, the Motion Detection has similar meaning with change detection, foreground detection and foreground/background extraction. By detecting the motion we can filter out the static background in scene, which meaning faster speed and higher accuracy in following computer vision tasks when we are only interested in moving objects. However, it is hard to distinguish the complex changes of real world scene from the motion of objects. In state of art method for Motion Detection, shadow of object, dynamic background and intermittent motion of objects remain challenging.

We use motion detection algorithm from the library offer by [**?**] and more detail about this library can be see from [**?**]. Considering the robustness and speed, we propose to use the algorithm in [**?**]. Our proposed tracking algorithm include follow steps:

1) Motion Detection: convert raw video to binary foreground mask video.
2) Identify objects in current frame: convert each binary mask into separated blobs through connect component algorithm.
3) Association objects in adjacent frames: use Local Feature Matching and Hungarian Algorithm to associate blob with tracked objects.
4) Create object association graph: combine two results from Local Invariant Feature Matching and Hungarian Algorithm, we classify objects and blobs into five types.
5) Occlusion and oversegmentation handle: according to the types and status of objects and blobs, use specify

strategies to deal with occlusion.

When there are few objects, the motion information is enough to tracking objects. However, when there are too many objects, it's easy to cause occlusion (object with scene and object with object)[**?**], [**?**]. And the more objects, the location and moving information become more ambiguous. In that case, the result of Hungarian algorithm mainly depends on the prediction for object's position. However, there are no way to predict the position of objects accurately because the moving under various environment is totally random. So we use local invariant feature to match blobs and objects and combine the results, which make tracking algorithm more robust. In order to generalize our tracking method, we do not distinguish car, bike, human and other objects for those work can also be done after tracking. Those objects different in size, speed, shape and color, which may also suffer from occlusion and oversegmentation, so the robust information remain only invariant local feature according to our knowledge, that's why we proposed to use invariant local feature to do objects split and merge. Our tracking algorithm based on motion detection, which is quiet different from the tracking algorithm based on feature based detection. We cannot suppose each detection result as one objects, and cannot directly locate the position of objects when occlusion happens; We also remove the limit of entry/exit area of image for the complexity of real application, for example, objects can come from building, which make it's extremely hard to specify an entry or exit area.

The highlight of our work: 1. we combine Hungarian algorithm with local invariant feature matching. 2. we use specify strategy to deal with occlusion and oversegmentation. The rest of this paper is organized as follows. Related work is discussed in section two. Our proposed multiple object tracking algorithm will be described in section three. Experiment will be included in section four and Conclusion will be given in section five..

II. Previous Work

In 2012, A new benchmark dataset CDNet (changedetection.net) for motion detection come out [**?**]. Based on this dataset it's easy to find out which algorithm is best for target motion detection application. Though there are many criteria and dataset to rank those algorithms before 2012, and the results may be different due to different dataset and different evaluation functions, however, N. Goyette offer a lot of means to rank those algorithms and the benchmark results in CDNet have been widely accepted. In 2014, the dateset add new video types according to the result and challenge researcher found on CDNet 2012[**?**]. So for a lot of computer vision application or research, it's easy to find a robust and effective algorithm in CDNet. What's more, for lots of algorithms, here is a motion detection source code library [**?**]. Those works give great convenience for followed video tasks. According to the experiment result, we propose to use the algorithm in [**?**], [**?**], [**?**], the first one is fast and open source, while the latter two algorithm need the author's authorization.

As for object tracking, a common approach is to do the association between detection results and objects frame by frame, in short frame sequence or a long frame sequence. Kalman filter [**?**] and particle filter[**?**] often combine motion information to tracking objects. For most of applications, Kalman filter or particle filter can help to do data association.

In paper [**?**]propose a maximum a posteriori (MAP) formulation to the multi-object tracking problem. This algorithm create a graph whose node is the observed blob and tracked objects, and use merge, split and mean shift operations to deal with the noise in motion detection. However the appearance feature use in edge weight is not enough for tracking objects when occlusion happened.

[**?**] proposed an intelligent transportation systems which target to automatically monitor the road traffic. The tracking algorithm is feature based, as an object have multiple feature point, so this tracking algorithm need to do feature point group after use Kanade-Lucat-Tomasi Feature Tracker. In this way, the model can resist to partial occlusion but the group problem remain to be study.

[**?**] proposed a real-time object tracking algorithm under fixed camera. This paper propose to do motion segmentation before object tracking, finally use object classification to obtain classified objects. The framework consider merging and splitting of objects, and use a priori assignment to do object merging and splitting. Though the motion segmentation algorithm and tracking algorithm is simple, but it's effective for some real-time video processing application according to their result.

[**?**] proposed min-cost network flow method, which can incorporate higher-order track smoothness constraints such as constant velocity. In this framework, the author apply Lagrangian relaxation to extra constraints. The node in this framework is the association between objects, and the extra constraints are used to punish conflicts between associations. The network flow cover the entire sequence, which mean this method is not feasible for online long object tracking.

[**?**] use the motion detection algorithm ViBe [**?**] from CDNet and modify ViBe according to the urban background to optimize the tracking result. Jean-Philippe Jodoin also find a solution for partial occlusion by using local invariant feature. They also offer video data and tools to help object tracking research in addition to tracking source code. Their experiment results show motion information can combine with local invariant feature to track object. Our algorithm will based on motion detection algorithm in [**?**] for the consideration of speed, using Kalman filter and Hungarian algorithm also help to improve our tracking speed, using local invariant feature to improve tracking robust. The experiment show our tracking algorithm is fast, robust and competitive to state-of-art tracking algorithm.

A. subsection

B. another subsection

III. Methodology

A. Overview

As we can see from Figure 1, Our algorithm follow the framework of "detection - association - tracking". Firstly we split the foreground mask into connect component(we call them foreground blobs later). Next we use Hungarian algorithm or local invariant feature matching to do association between foreground blobs and tracked objects. Next we create the association graph for blobs and tracked objects, we classify node in association graph. For nodes need merge or split, we handle the occlusion and oversegmentation for them; For nodes need new or delete, we update the tracked objects; For nodes need update, we maintain the tracking status for them. Finally, we output the trajectory result for current frame.

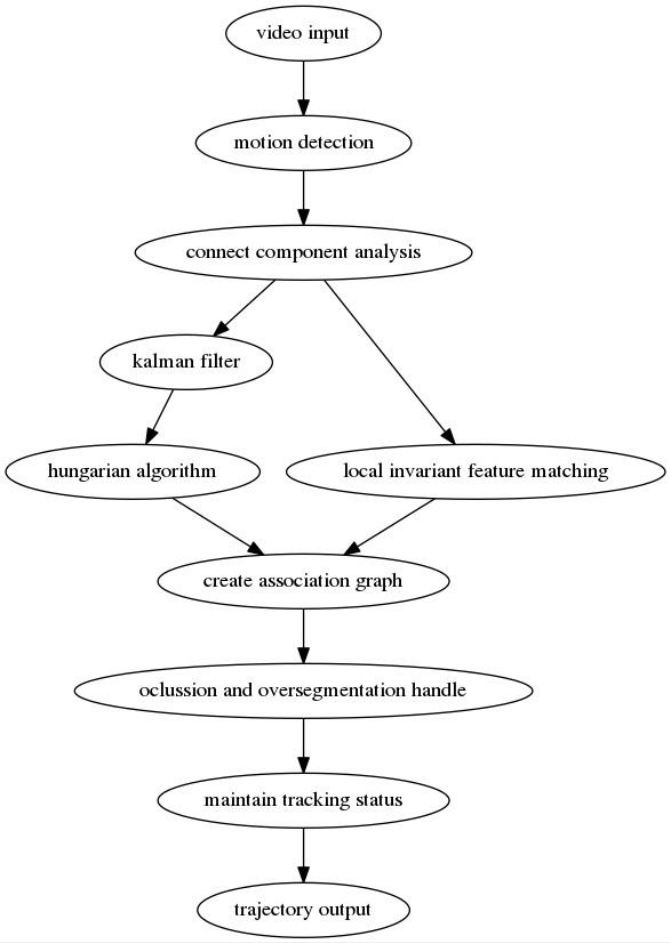

Fig. 1. Our Tracking Algorithm Overview

B. Motion Detection

Figure 2 show basic framework of Motion Detection based background model. When $t < 1$ we initialize the background model, then for the rest of $t \geq 1$ we update the background model according to input frame and output motion mask. The mask is generated from motion response image by a threshold and the motion response image is calculated by the distance between input frame and background model.

Mathematical Symbol	Meaning
t	frame number, the timestamp for input video,
$I^t_{input}(i,j)$	pixel at location (i,j) in input frame at time t
I_{mask}	image: mask for motion object area
C_{width}, C_{height}	width and height of input frame I_{input}
$I_{response}$	image: response of Motion Detection
O^t_i	the ith Object in I^t_{input} (include the missed objects in previous input frames)
B^t_i	the ith Blob in I^t_{input}(only blobs in current input frame)
N^t_{object}, N^t_{blob}	objects and blobs Number in I^t_{input}

TABLE I
Mathematical Symbol and Meaning Overview

We have $I_{response} = f_{distance}(I_{input}, B_{model})$, background model B_{model} may not just an image, for motion detection algorithm like VIBE, PBAS,SUBSENSE. B_{model} contains 10~50 background images. Distance function $f_{distance}$ compute the distance between current input image and background model. And we have:

$$I_{mask}(i,j) = \begin{cases} 0 & I_{response}(i,j) > C_{response} \\ 1 & I_{response}(i,j) \leq C_{response} \end{cases}$$

$C_{response}$ is the threshold for motion response image $I_{response}$. According the the benchmark and survey[**?**], we choose MultiCueBGS [**?**] as our motion detection method for it's advantage on speed and it's good resistance to background noise, light change.

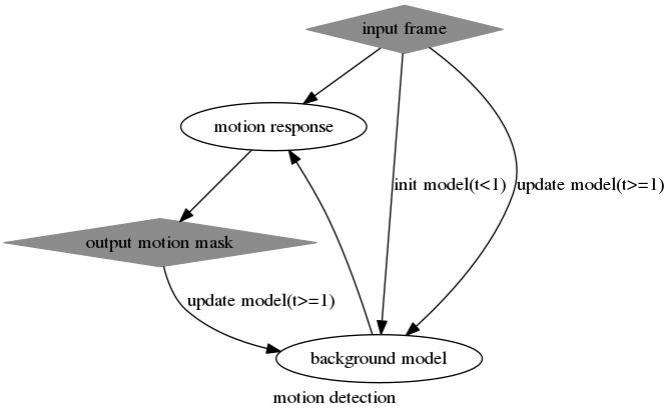

Fig. 2. Motion Detection

C. Identify Objects In Current Frame

Because there may be two or more objects in I^t_{input}, we need to identify each object alone. By use Connect Component Analysis algorithm we can get N^t_{blob} connect component in

I^t_{input} and due to the false negative and false positive in I^t_{mask}, the number of connect component may not equal to the number of objects N^t_{object}. Two or more blob may come from the same object and one blob may contain two or more object too. In addition, there are pure noise blob as well. We use a area threshold C_{area} to remove small blob and extract the center position for each remain blob.

D. Association Objects in Adjacent Frames

We have to assign blobs to objects, which can be see as an assignment problem. If we suppose one blob only corresponds to one object, we can use Hungarian Algorithm to do this. However, above suppose is not true for most of time and we need to assign two or more blobs to one object and assign two or more objects to one blob. So we use local invariant feature to extend the Hungarian algorithm and deal the situation of object merge and split.

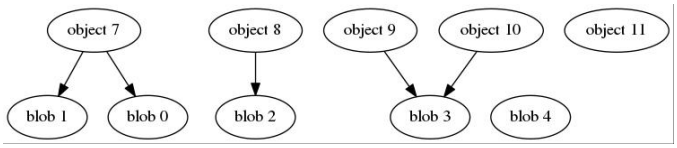

Fig. 3. Assignment example

As we can see from Figure 3, the assignment between objects and blobs have 5 type: split, merge, normal, miss and new; object 7 split into blob 1 and blob 0, object 8 normally matched to blob2, object 9 and object 10 merged into blob3, object 11 and blob 4 have no matched blob/object. For object we called it's missing and for blob we need to new an object to match it. For a long period video tracking, the object may disappear.

E. Create Object Association Graph

Though use Hungarian Algorithm alone cannot deal with the problem of split and merge. However in most of time, Hungarian Algorithm can do right thing and we proposed to deal with split and merge alone with invariant local feature. For objects $O^t_i, i \in [1, N^t_{object}]$ and blobs $B^t_i, i \in [1, N^t_{blob}]$ we get the cost matrix:

$$\begin{bmatrix} d & diag(d^{new}) \\ diag(d^{miss}) & 0 \end{bmatrix}$$

$$= \begin{bmatrix} d_{11} & \cdots & d_{1N} & d^{new}_1 & \cdots & +\infty \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ d_{M1} & \cdots & d_{MN} & +\infty & \cdots & d^{new}_M \\ d^{miss}_1 & \cdots & +\infty & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ +\infty & \cdots & d^{miss}_N & 0 & \cdots & 0 \end{bmatrix}$$

here d is the distance matrix for O^t_i and B^t_i where $d_{ij} = distance(B^t_i, O^t_j)$, and $M = N^t_{blob}, N = N^t_{object}$. The $d^{miss}_i, d^{new}_j, i \in [1, N], j \in [1, M]$ are the cost for a missing object i or the cost for a new object to match blob j. Use Hungarian Algorithm to address the assignment problem we can get the blobs need to match new objects, the objects missed and the normal match information between the objects and blobs. However, Hungarian Algorithm cannot obtain the occlusion information, which is a common thing often happened in tracking system. Normally, there are three ways to obtain the occlusion information when combine Local Feature Matching and Hungarian algorithm. For blobs which need matching new objects, we use local invariant feature to find whether it is the split of existed object.

1) Using Local Feature Matching first, we can obtain the association between blobs. Because each blob may two or more blobs which have enough matched feature points with it, we can obtain occlusion information from Local Feature Matching alone. However, for smooth object or small object which has few feature points, it often does not has enough feature points, so use Local Feature Matching alone, the smooth object and small object cannot be tracked well. So for missed objects or the blobs unmatched after Local Feature Matching, we use Hungarian algorithm to tracking them. This strategy can be see a

$$\begin{aligned} G^t_{LFM} &= LFM(O^t, B^t) = (V^t_{LFM}, E^t_{LFM}) \\ O^t &= \{O^t_i | i \in [1, N^t_{object}]\} \\ B^t &= \{B^t_i | i \in [1, N^t_{blob}]\} \\ B^t_{\neg LFM} &= \{v | v \in B^t, \forall u \in O^t, < u, v > \notin E^t_{LFM}\} \\ O^t_{\neg LFM} &= \{u | u \in O^t, \forall v \in B^t, < u, v > \notin E^t_{LFM}\} \\ G^t_{HA} &= HA(O^t_{\neg LFM}, B^t_{\neg LFM}) = (V^t_{HA}, E^t_{HA}) \end{aligned}$$

O^t is the object set and B^t is the blob set for frame I^t_{input}, E^t_{LFM} is the matches of Local Feature Matching, we map each matches as edge in our association graph. V^t_{LFM} is the vertex for graph. G^t_{LFM} is the association graph for Local Feature Matching and G^t_{HA} for Hungarian Algorithm. $B^t_{\neg LFM}$ and $O^t_{\neg LFM}$ are the remain blobs and objects, we use them as input for Hungarian Algorithm which denote as $G^t_{HA} = HA(O^t_{\neg LFM}, B^t_{\neg LFM})$.

2) To combine the motion information and local invariant feature information for all objects and blobs, different from strategy above, we can match all objects and blobs in Hungarian algorithm. This strategy will be helpful when the feature points matching is not robust and we add edge for the final graph, which mean a more complex model for occlusion. In math format we have:

$$\begin{aligned} G^t_{HA} &= HA(O^t, B^t) = (V^t_{HA}, E^t_{HA}) \\ G^t_{LFM} &= LFM(O^t, B^t) = (V^t_{LFM}, E^t_{LFM}) \end{aligned}$$

3) For case where motion information is more robust than feature matching, for example, many car of the same type. We can use Hungarian algorithm first and then use feature point matching for the rest, in this case, we

have:

$$G^t_{HA} = HA(O^t, B^t) = (V^t_{HA}, E^t_{HA})$$
$$G^t_{LFM} = LFM(O^t_{\neg HA}, B^t_{\neg HA}) = (V^t_{LFM}, E^t_{LFM})$$

Finally, the association graph is the combine of G^t_{HA} and G^t_{LFM}, in math format is:

$$G^t = G^t_{HA} \cup G^t_{LFM} = (V^t_{HA} \cup V^t_{LFM}, E^t_{HA} \cup E^t_{LFM})$$

When the assignment results for both method is high accuracy but low recall, we should choose the combination of 2. When the assignment results for Local Invariant Feature matching is low accuracy but high recall, while the assignment results for Hungarian algorithm is high accuracy but low recall, we should choose the combination of 3. In our implementation, we ensure the Local Invariant Feature matching with high accuracy. First we filter the matches with ratio test and symmetry test according to [**?**]. Second we filter out the matches with distance radius which bigger than a threshold $T_{maxMatchDistance}$, this avoid unexpected matches with two faraway feature points. Finally, we do Hungarian algorithm on rest unmatched objects and unmatched blobs. We use only motion information for Hungarian algorithm, without color and appearance information. Thus we can match objects when their color and appearance change while be tracked. Use motion information only make Hungarian algorithm with low accuracy but high accuracy. So we choose the combination of 1 for the final assignment.

F. Handle Occlusion

For those unmatched object and blob, we local invariant feature to match them. If the matched point more than 3, we accept that match and else we reject it. As we can see from Figure 4, the edge label "LIF" mean Local Invariant Feature matching for object and blob, "Hun" mean Hungarian algorithm matching. Then we will store the status of merge object 2 and object 3. However, the object 5 split into blob 3 and blob 4, and we will create new object 6 and object 7 to match blob 3 and blob 4. The ways to handle each objects and blobs depends on their type and status, we classify the association result as follow type:

1) Blobs without associated objects

$$A_1 = \{v | v \in B^t, \forall u \in O^t, < u, v > \notin E(G^t)\}$$

2) Objects without associated blobs.

$$A_2 = \{u | u \in O^t, \forall v \in B^t, < u, v > \notin E(G^t)\}$$

3) Objects associated with one and only one blob.

$$O(u) = \{u | \exists v \in B^t, < u, v > \in E(G^t)\}$$
$$A_3 = \{u | \, \|O(u)\| = 1\}$$

4) Objects associated with two or more blobs.

$$A_4 = \{u | \, \|O(u)\| > 1\}$$

5) Blobs associated with two or more objects.

$$B(v) = \{v | \exists u \in O^t, < u, v > \in E(G^t)\}$$
$$A_5 = \{v | \, \|B(v)\| > 1\}$$

For association type of A_1 we create new objects to track, we denote those new objects as $New(A_1)$, operation on A_1 convert blobs to objects; For A_2 we update the tracking status for them, for objects which cannot get associated blobs for long time ($> T_{maxMissTimes}$), we will remove them. For convenience we denote those delete objects as $Delete(A_2)$; For A_3 we just update the tracking status for them; For A_4:

1) If a blob have enough distance $T_{splitDistance}$ with other blobs, we split it out. This is the case for many objects with different destinations. However, we need notice the distance threshold is depend on the size of objects, the angle of camera, the speed of objects, the frame rate of video, the size of frame and the distance between camera and objects. And in current dataset MOT2015, the annotation shape of objects and blobs is rectangle only, so the shape can also have influence on the distance threshold.
2) If two blobs have been split for enough times $T_{splitTimes}$, we split them. We avoid use the split time to split two blobs but times. Because it's often the case that an objects linger on background cluster, which cause the false negative for foreground mask and the split of objects. To avoid this problem, we prefer to use the split times while objects is moving than to predict the lingering time on background cluster for objects. A case for this is two friend go together, the foreground mask of them will split and merge often when they are moving.
3) If two or more blobs come from faraway to nearby, the foreground mask for them may split. To address this problem, we need to detect the size change of objects, when the size of objects increase with time, and finally the objects associated with two or more objects, we split them out.
4) For all of the blobs split out, we set their status to avoid them merge with the object that associated with them. This strategy is mean to avoid split and merge loop.

For A_5, we merge only when:

1) The objects are fresh. This mean we do not merge objects for long time, and we prefer to use the lifetime of objects rather than the position of objects to filter objects which can be merged. Because it's often the case that objects can come from area which is not the border of frame.
2) The objects are not marked by split operation, as we mentioned above, we avoid split and merge loop.
3) The object's trajectory can be merged. This mean the objects should stay closed enough from the first time two objects both appeared in frames.

As mentioned above, we denote the split object in A_4 as $Delete(A_4)$ and the split blobs in A_4 as $New(A_4)$, the merged

blobs in A_5 as $New(A_5)$ and the merged objects in A_5 as $Delete(A_5)$. Finally we have:

$$O^{t+1} = Update(O^t) + New(A_1) - Delete(A_2) + New(A_4) - Delete(A_4) + New(A_5) - Delte(A_5)$$

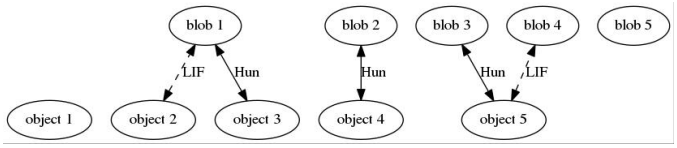

Fig. 4. Association type example: A_1={blob 5}, A_2={object 1},A_3={object 4},A_4={object 5},A_5={blob 1}.

IV. Results

A single column figure goes here

Fig. 5. Captions go *under* the figure

TABLE II
TABLE CAPTIONS GO *above* THE TABLE

delete	this
example	table

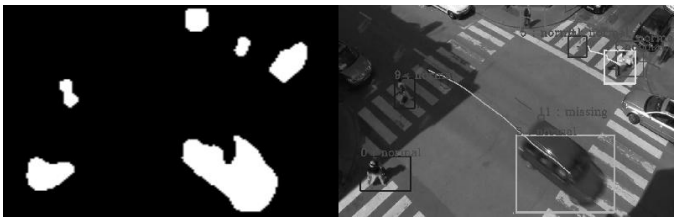

Fig. 6. Motion detection result and tracking result of our proposed algorithm, we can see occlusion for people on zebra crossing, we split the blob correctly and for other small object the Hungarian algorithm's assignment is acceptable. But the noise from motion detection remain to be removed for the static blob on zebra crossing and there are false negative on car.

We run our algorithm on Urban Tracker Dataset[**?**], we can see our result in Table III. Note we do not use the Sherb and Ren-L dataset because those two dataset use mask to remove noise area for motion detection area. To compare those two algorithm in the same level, we choose the same motion detection algorithm LOBSTER [**?**]. In our experiment we find LOBSTER is slow and MultiCueBGS [**?**] have the competitive result with LOBSTER but faster speed. So we proposed to use MultiCueBGS as the motion detection algorithm. The feature detect algorithm and extract algorithm are BRISK [**?**] and FREAK [**?**], which is proposed in Urban Tracker. The two main evaluation metric are Multiple Object Tracking Precision(MOTP) and Multiple Object Tracking Accuracy(MOTA) in [**?**]. Smaller MOTP and larger MOTA mean better result.

video	Urban Tracker		Our Method 1		Our Method 2	
	MOTA	MOTP	MOTA	MOTP	MOTA	MOTP
Atrium	0.59	77.4	0.65	70.3	0.62	70.0
Rouen	-0.18	71.6	0.44	69.4	0.43	69.1
st-Marc	0.23	73.4	0.53	63.4	0.51	63.2
average	0.21	74.1	0.54	67.7	0.52	67.4

TABLE III
TRACKING EVALUATION ON URBAN TRACKER DATASET, THE URBAN TRACKER HERE USE LOBSTER AS THE MOTION DETECTION ALGORITHM IN TRACKING, WHICH IS THE PROPOSED ALGORITHM FROM IN SOURCE CODE. OUR METHOD 1 USE LOBSTER BUT OUR METHOD 2 USE MULTICUEBGS.

parameter	value	detail
D_i^{miss}, D_i^{new}	200	the distance threshold for Hungarian Algorithm
$T_{maxMissTimes}$	10	the max miss times for objects before removed
C_{area}	100	the area threshold to remove small noise foreground mask
$T_{splitDistance}$	100	the distance threshold to split objects
$T_{splitTimes}$	3	the times threshold to split objects
T_{fresh}	5	the lifetime threshold to determine wheater a object is fresh when merge them
$T_{maxMatchDistance}$	100	the max distance for matched feature points

TABLE IV
OUR METHOD'S PARAMETER

V. Conclusions

We proposed a motion detection based tracking system to tracking all kinds of object under various environment in the same framework. Experiment result show that our algorithm is more robust than current state-of-art tracker. We combine Hungarian Algorithm and Local Feature Matching to offer robust tracking result, use mark and novel status to distinguish split blobs from merged blobs. Both motion detection error and occlusion can be solved by our method and our experiment result prove it.

Appendix A
First appendix

Citation: [1]

Appendix B
Second appendix

Acknowlegment

bla bla

References

[1] N. H. F. Beebe. (2010, Dec.) TEX user group bibliography archive. [Online]. Available: http://www.math.utah.edu/pub/tex/bib/index-table.html