MAPTrack
A Probabilistic Real Time Tracking Framework by Integrating Motion, Appearance and Position Models

Saikat Basu[1], Manohar Karki[1], Malcolm Stagg[2], Robert DiBiano[1],
Sangram Ganguly[3] and Supratik Mukhopadhyay[1]
[1]*Department of Computer Science, Louisiana State University, 102F Electrical Engineering Building, 70803, Baton Rouge, Louisiana, U.S.A.*
[2] *Remote Desktop Virtualization team, Microsoft Corporation, One Microsoft Way, 98052, Redmond, Washington, U.S.A.*
[3] *Biosphere Science, Bay Area Environmental Research Institute/NASA Ames Research Center, Building - 19, NASA Ames Research Center, 94035, Moffett Field, California, U.S.A.*

Keywords: Object Tracking, Motion Model, Appearance Model, Gaussian Mixture Background Subtraction, Optical Flow.

Abstract: In this paper, we present MAPTrack - a robust tracking framework that uses a probabilistic scheme to combine a motion model of an object with that of its appearance and an estimation of its position. The motion of the object is modelled using the Gaussian Mixture Background Subtraction algorithm, the appearance of the tracked object is enumerated using a color histogram and the projected location of the tracked object in the image space/frame sequence is computed by applying a Gaussian to the Region of Interest. Our tracking framework is robust to abrupt changes in lighting conditions, can follow an object through occlusions, and can simultaneously track multiple moving foreground objects of different types (e.g., vehicles, human, etc.) even when they are closely spaced. It is able to start tracks automatically based on a spatio-temporal filtering algorithm. A "dynamic" integration of the framework with optical flow allows us to track videos resulting from significant camera motion. A C++ implementation of the framework has outperformed existing visual tracking algorithms on most videos in the Video Image Retrieval and Analysis Tool (VIRAT), TUD, and the Tracking-Learning-Detection (TLD) datasets.

1 INTRODUCTION

Tracking moving objects in a streaming video in real time is important for many applications such as video surveillance, activity recognition, robotics, etc. A statistical method for parametric modeling of object geometry as well as illumination changes owing to variance in lighting conditions was proposed in (Hager and Belhumeur, 1998). However, their approach was only used particularly in tracking human faces; no results are available for videos involving objects having different types of motion such as vehicles, humans, etc. that interact with each other (closely) as is often the case in surveillance videos. Moreno-Noguer et al (2008), provide a Bayesian framework for combining information obtained about appearance and object geometry for robust visual tracking. However, their framework cannot track multiple moving objects simultaneously; in addition, it cannot handle occlusions.

In this paper, we present a probabilistic real time tracking framework that combines the motion model of an object with its appearance and position. The motion of the object is modeled using the Gaussian Mixture Background Subtraction algorithm, the appearance, by a color histogram and the projected location of the tracked object in the image space/frame sequence is computed by applying a Gaussian to the Region of Interest. Our tracking framework is robust to abrupt changes in lighting conditions, can follow an object through occlusions, and can simultaneously track multiple moving foreground objects of different types (e.g., vehicles, human, etc.) even when closely spaced. A spatio-temporal filtering algorithm helps in automatic track initialization and a "dynamic" integration of the

Basu S., Karki M., Stagg M., DiBiano R., Ganguly S. and Mukhopadhyay S..
MAPTrack - A Probabilistic Real Time Tracking Framework by Integrating Motion, Appearance and Position Models.
DOI: 10.5220/0005309805670574
In *Proceedings of the 10th International Conference on Computer Vision Theory and Applications* (VISAPP-2015), pages 567-574
ISBN: 978-989-758-091-8

framework with optical flow allows us to track videos with significant camera motion. A C++ implementation of the framework has outperformed existing visual tracking algorithms on most videos in the Video Image Retrieval and Analysis Tool (VIRAT), TUD (Andriluka et al, 2008), and the Tracking-Learning-Detection (Kalal et al, 2010) datasets.

2 RELATED WORK

A new particle filter - Kernel Particle Filter (KPF), was proposed in the (Chang and Ansari, 2005) for visual tracking for multiple objects in image sequences. The idea proposed in (Williams et al, 2003) shows tracking using a single classifier SVM. A boosting based approach was proposed in (Viola and Jones, 2001) that used a cascade of classifiers for object detection. However, it didn't address the problem of tracking objects through consecutive frames of a video sequence. A spatio-temporal tracking algorithm was proposed in (Lan and Huttenlocher, 2004) that involved tracking articulated objects in image sequences through self-occlusions and changes in viewpoint. However, they did not provide capabilities for automatic track initialization or tracking multiple objects.

The TLD algorithm proposed in (Kalal et al, 2010) is the basis of one of the well-known frameworks for tracking moving objects. The TLD framework does not start tracks automatically; it lacks a multi-object tracking feature. Also, TLD is based on template matching and hence fails for videos with multiple numbers of similar looking objects (e.g., in the Indian driving scene video, Figure 5). The approach proposed in (Maggio and Cavallaro, 2005) uses color histograms as the only feature. They use a cascade composition of a particle filter and mean shift. The method proposed in (Babenko et al, 2009) is similar to the approach proposed in TLD. The difference between the work reported in it and TLD is that they use multiple instances as the positive examples in each frame. However, like TLD, their framework does not start tracks automatically as marking the location of the object initially is a prerequisite. A Bayesian estimation-based object tracking algorithm that takes into account the motion models, shape and appearance constraints has been proposed in (Tao et al, 2002) but it fails when the motion layers are infiltrated with clutter, occlusion etc. Another method for detecting event sequences in surveillance videos that is applicable only to low frame rate videos is proposed in (Lombardi and Versino, 2011).

Our approach is based on using the motion model, color histogram, and position information of objects to track them with a recursive probabilistic estimation of the composite model. Unlike the previous approaches, it can simultaneously track multiple moving objects, does not fail significantly when there is no motion, or when the object is occluded, is resistant to clutter, and is also able to initialize tracks without human supervision.

Recently, there have been a lot of works that combine multi object tracking, multi-person tracking, and association between different tracked objects for activity recognition (Oltramari and Lebiere, 2012). Our framework tracks multiple objects in a video in each frame or multiple frames efficiently; this capability could be used as a part of a co-related and collective activity recognition framework.

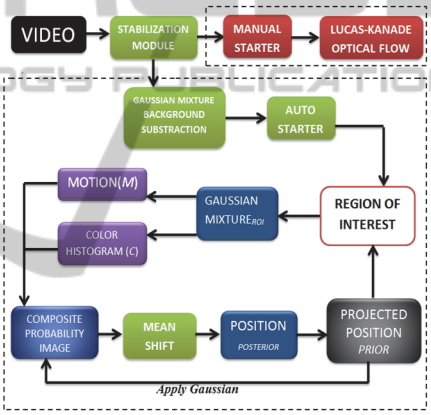

Figure 1: Schematic representation of our approach.

3 THE PROPOSED APPROACH

Figure 1 shows the schematic of our approach. First, a moving object must be automatically identified as part of the foreground. This involves track initialization at particular pixels on the subsequent frames that have a higher probability of being part of the moving foreground object. This is achieved by - 1) stabilizing the image and 2) feeding the stabilized image to the spatial and temporal filtering algorithms described below. Issues such as camera instability (shaking, panning, rotating) come into play and require image stabilization for the tracking. These issues and the components of the tracking framework are described in detail below.

3.1 Image Stabilization

In order to stabilize an incoming streaming video, we use an iterative algorithm which attempts to hold each background pixel in the same position regardless of lateral and rotational camera motion:

1. Apply Shi and Tomasi's edge-finding algorithm to the first frame to identify significant feature points in the image.
2. For each subsequent frame, apply Lucas-Kanade optical flow to track the motion of the features identified by Shi and Tomasi's algorithm, refreshing the feature points when necessary.
3. With increasing precision for each iteration:
   (a) For each angle of rotation in a certain range, determine the translation of each point.
   (b) Find the most common translation/rotation (mode) pair (Θ, x) and (Θ, y) of all the features.
4. Warp the image to adjust for the total mode of the motion.

Before adjusting for background motion, we must identify features of the frame; to do so, we use the Shi-Tomasi method (Shi and Tomasi, 1994). The Shi-Tomasi method detects features such as corners and edges by approximating the weighted sum of squares of image patches shifted by certain values.

Next, we apply a pyramidal Lucas-Kanade method (Lucas and Kanade, 1981) for determining optical flow at each point of interest. We then find the mode of the resulting flow value pairs, including rotation, by placing the pairs in bins. At every iteration, the bin widths are decreased, yielding an increasingly accurate estimate of the motion. The image is then adjusted to account for the determined background movement. When the image is stabilized in this manner, fewer false foreground objects detected and correct coordinates of objects are also maintained.

3.2 Automated Track Initialization

The automated track initialization algorithm based on a confidence-based spatio-temporal filtering algorithm first detects blobs using the GM Background Subtraction method (KaewTraKulPong and Bowden, 2002). This yields difference images, which are fed into the spatial filtering module.

3.2.1 Noise Removal through Morphological Operations

The image obtained through the background subtraction algorithm is initially opened and then closed by a structuring element with diameter λ pixels to filter out unnecessary noise. λ depends upon the scale of the video.

3.2.2 Spatial Filtering

Once blobs are detected in the difference images, they are filtered according to their spatial features. Scale information available from the metadata accompanying the videos is used to filter blobs specifically based on their area and orientation. The filtered blobs are then passed as input to the temporal filtering algorithm below.

3.2.3 Temporal Filtering

To filter blobs in the temporal domain we use a *confidence measure*. Each blob has a confidence measure δ associated with it. δ is initially 0 and increases as a blob is detected across successive frames.

The probabilistic framework that we present takes into account three parameters namely, the motion of the object modeled using the Gaussian Mixture Background Subtraction algorithm, the appearance of the tracked object, using a color histogram, and the projected location of the tracked object in the image space/frame sequence computed by applying a Gaussian to the Region of Interest.

3.2.4 Defining an Adaptive Threshold

If the confidence value for a blob exceeds a specified upper threshold σ, a track is started on it. The moment the confidence value for a blob falls beneath a lower threshold τ, the corresponding object is discarded. If the confidence value is between σ and τ, the corresponding blob is maintained in the list of prospective tracks. For videos that have higher noise, clutter and random changes in lighting conditions, as is often the case for outdoor videos taken from moving cameras, the upper threshold value σ is set higher. On the other hand, for videos with more stable conditions σ is set lower because of the lesser probability of encountering random classification noise.

The composite confidence update equation is as follows:

$$\delta = (0.5^{-n}) \vee (-0.5^{-n}) \quad (1)$$

3.3 The MAPTrack Framework

Motion – The Gaussian Mixture Background subtraction method helps in determining the positional estimates for all *moving* objects in the

scene. It is reasonable to consider all *moving* objects to be a part of the foreground. Our framework builds a background model of Gaussians, with a mean and standard deviation for each pixel. If a new pixel does not fit well with the Gaussians, it is considered to be part of the moving foreground.

Appearance – The appearance of any object in a scene is another important parameter in visual tracking. Object appearance can be modeled using the color histogram associated with it. Operationally, the motion image is used as a mask to create histograms of object pixels for each ROI. Histograms are implemented as 3D RGB histograms with 32 bins in each R, G, and B direction. For example, bin(0,0,0) contains R=0 to 7, G=0 to 7, B=0 to 7, etc.

Histograms are created for foreground (h_{fg}) and background (h_{bg}) components of the current motion image at the current frame. Each bin in a current histogram contains the count of the pixels, which fall in that bin. The background histograms are normalized to make the count of pixels in each equal to the number of foreground pixels in the motion image:

$$h_{bg} = \frac{h_{bg} \times |h_{fg}|}{|h_{bg}|} \tag{2}$$

So, both foreground and background image have magnitude equal to the number of foreground pixels in the motion image (e.g. when the object is stopped, both current-frame histograms have 0 magnitude). The cumulative histograms (H_{fg} and H_{bg}) are updated using a running average:

$$H = \frac{\acute{H} \times (n-1) + h}{n} \tag{3}$$

where, n is minimum of the current frame number and the point at which the average will change to exponential decay and $\acute{H}$ is the cumulative histogram value from the last frame.

A probability image is created for the pixels in the ROI from the Bayes equation:

$$P\,(FG|\hat{z}) = \frac{P(\hat{z}|FG) \times P(FG)}{P(\hat{z})}$$
$$= H(\hat{z}) \times \frac{avgFG}{(H(\hat{z}) \times avgFG + (H(\hat{z}) \times (1 - avgFG)}$$
$$P(x,y) = 1 \text{ if } P(FG|\hat{z}) > 0.5, \text{ else } 0 \tag{4}$$

where, avgFG is the sum of the motion history image described below. In other words, if a pixel color is more likely to lie in the object foreground, it will be '1'. Otherwise it will be '0'.

Projected Position – The estimate of the projected position of an object over an image sequence is another determining factor in visual tracking. The position is estimated using the previous position and estimated velocity:

$$p_{est} = \acute{p}_{act} + v \tag{5}$$

where, v is calculated as:

$$v = \frac{p - p_0}{f - f_0} \tag{6}$$

Here, f_0 is the nearest previous frame where the object is at a distance of at least 1 ROI width from current position if it exists and $max(0, f\text{-}150)$, otherwise. p_0 is the position at that frame.

A positional probability image for the ROI is created using a conical shape.

$$Prob(x,y) = max - (max - min) \times \sqrt{\left(\frac{x - c_x}{c_x}\right)^2 + \left(\frac{y - c_y}{c_y}\right)^2} \tag{7}$$

where, (c_x, c_y) is the center of the ROI, max is the probability value at the center that is equal to 1, and min is the probability at the edges.

Figure 2: a) R b) Motion Image c) Color Hist. Image d) Position Image.

This image represents the estimated position or velocity of the object, and reduces movement from this estimated location. Thus, the probability is highest where the object is most likely to be present (in the center).

In addition, a motion history image is created to estimate the probable object shape, size, and location within the ROI. Similar to the color histograms, it is updated as:

$$mh(f) = \frac{(mh(f-1) \times historysize(f-1) + MC \times w)}{(historysize(f-1) + w)} \tag{8}$$

where, $historysize(f) = min(historysize(f-1) + w, N)$, and $w = \sum_{i=1}^{T}(\frac{MC}{ROI_{area}})$ as a scale factor based on the amount of movement present. N is again the point at which the average will change to exponential decay. MC is the image of all moving pixels in the ROI matching the foreground color, as determined by the color histogram, of the object. T represents all the pixels in ROI. Each of the Motion,

Color, and Positional probability images is centered over the estimated position calculated above. Once the images are obtained, they are combined into a *composite probability image (CPI)* by using the following equation:

$$CPI = max(M \times C \times P \times \sigma_1, C \times P \times \sigma_2, P \times \sigma_3) \quad (9)$$

Here, σ_1, σ_2 and σ_3 are parameters that determine the relative weights given to the Motion, Color histogram, and Positional Probability images respectively towards the Composite Probability Image *I*. Intuitively, equation 9 is a recursive OR over the values *M×C×P*, *C×P* and *P* where the *C×P* or *P* parts are used only when the *M* value is 0 and *P* is used only when both *M* and *C* are 0. It should be noted that *M×C×P* uses the conical probability image for *P*, to utilize any motion of matching color within the ROI, whereas *C×P* uses the motion history image for P, such that still background of a matching color will not cause a track loss.

Since the motion probability image is the most important parameter for object tracking, *MCP* is assigned the highest weight. The color histogram probability image is less important, followed by the positional probability image. In fact, we found that the *P* image alone does not work well to deal with occlusions due to the effective velocity of the object decreasing immediately before an occlusion.

The occlusion detection algorithm described below is instead used to handle occlusions and changing lighting conditions. Finally, the mean shift algorithm is used to compute the actual position of the object by shifting the estimate to the new Center of Mass (COM) of the current observation. The mean shift equation is given in equation 10.

$$Pos_{act}(f) = Pos_{est}(f) + COM(f) \quad (10)$$

where, $Pos_{act}(f)$ is the actual position computed at frame f, $Pos_{est}(f)$ is the estimated position at frame f and $COM(f)$ is the Center of Mass used by the mean shift algorithm for estimating the actual position of the object at frame f.

So, the mean shift gives the posterior probability distribution given the prior and the likelihood function. The positional estimate for the actual object location generated by the mean shift algorithm for a given frame f is used to compute the positional estimate for the next frame $f+1$ according to equation 5 and the system continues.

Occlusion Detection – The problem with using the position probability image (*P*) to handle an occlusion was primarily due to the decreasing effective velocity (since the occluded edge is not effectively moving, the velocity of the center of mass effectively reduces) of the object prior to the occlusion since the partially occluded center of mass moves at approximately half of the actual velocity of the object. Since *P* would only be used where *M×C×P* and *C×P* are very small, a metric is instead used to detect an occlusion:

$$occval = \frac{\sum_{i=1}^{T}(CP_{motionhistory})^2}{\sum_{i=1}^{T} C} \quad (11)$$

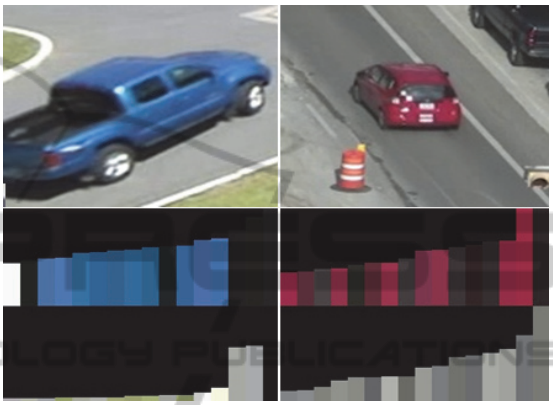

Figure 3: Foreground and Background Color Histograms of the two cars.

$CP_{motionhistory}$ is the estimate of the amount of color matching in the object foreground, and C is the amount of matching color in all of the ROI. Thus, *occval* will be small when either the amount of color in the object is small, or the amount of matching color in the background is very large. An occlusion is detected if:

$$occval(f) < t_{occ} \times max(occval(n)) \quad (12)$$

where, n is a frame number between f_0 and f with f being the current frame and f_0 the nearest frame where the object is at least one ROI width distance from the current position, or max(1, f-150) if that doesn't exist, and t_{occ} is the threshold for occlusion. When an occlusion is detected, the velocity from frame f_0 is used as an estimate of the current velocity, and the current position is adjusted to reflect that velocity:

$$p_{est}(f) = p_{est}(f_0) + v(f_0) \times (f - f_0) \quad (13)$$

This estimated velocity remains the same while the object is occluded. The ROI is allowed to drift up to half its length from the estimated position towards the center of mass while occluded to allow it to jump to the object when it is again present. The occlusion is ended when significant motion of matching color is again present:

$$\sum_{i=1}^{N} MCP \times \left(\frac{\sum_{i=1}^{T} CP_{motionhistory}}{\sum_{i=1}^{N} C}\right) > \tau * max(occval(n)) \quad (14)$$

where, τ is the threshold to end the occlusion, currently set to 0.3. If the occlusion does not resolve within 150 frames of 3 ROI widths, whichever is smaller, the track is ended.

4 IMPLEMENTATION DETAILS

The tracking algorithm was implemented in C++ using the OpenCV library for real-time computer vision. The experiments were conducted on an Intel I5 machine with 6 gigabytes of memory.

Figure 4: Output from MAPTrack (Left) and TLD (right).

5 RESULTS AND COMPARATIVE STUDIES

We compare results from our tracker against existing trackers whose outputs are available at the publicly available TLD dataset (Kalal et al, 2010). Table 1 shows the different states of the tracked object inferred at different values for the Motion, Color and Positional Probability images. Table 2 gives the number of frames up to the first track loss for the TUD dataset (Andriluka et al, 2008). It can be seen that MAPTrack outperforms the TUD Detector on both categories of the TUD Dataset. Table 3 shows the number of frames after which the trackers lost track for the first time. MAPTrack outperforms other trackers in all of the cases (except motocross). TLD is based on template matching and hence fails for videos with multiple similar looking objects. This is illustrated in Figure 4 where TLD switches tracks arbitrarily between similar looking foreground objects whereas MAPTrack keeps tracking a particular object for the entire time frame of its visibility. We also compare our tracker against the TUD Pedestrian Detector (Andriluka et al, 2008) for multi-object tracking. The performance metric used was taken from in (Smith et al, 2005). Figure 7 shows the ROC curve for the tracker and Figure 8 shows the results from MAPTrack. Table 4 lists the results for occlusion on videos from the VIRAT public dataset available online (VIRAT).

Table 1: The different states of the tracked object.

Motion	Color Histogram	Projected Position	Inferred State
0	0	0	Lost Track
0	0	1	Occlusion
0	1	0	Wrong Object
0	1	1	Stopped
1	0	0	Wrong Object
1	0	1	Wrong Object
1	1	0	Wrong Object
1	1	1	Moving Object

Figure 5: MAPTrack results for TUD videos.

Table 3: Comparison of single-object trackers in (Kalal et al, 2010) with MAPTrack.

Algorithms	Jumping (frames=313)	Car (frames=45)	Motocross (frames=2665)	Car chase (frames=9928)	Panda (frames=3000)
Beyond semi-supervised	14	28	6	66	130
Co-trained Generative-Discriminative	11	34	1	1	1
"CVPR" results	96	29	59	334	358
Online Multiple Instance Learning	313	220	63	321	992
Online Boosting	26	545	-	-	-
Semi-Supervised On-line Boosting	21	652	59	190	83
TLD	313	802	173	244	277
MAPTrack	**313**	**821**	**162**	**402**	**2568**

Table 4: Results from the tracker (Metric used as in (Smith et al, 2005)).

Video	Duration	CD	MO	MT	FP	TP	Occlusions	TP	FP
VIRAT_S_010000_01_000184_000324	1m 49s	1.3823	0.036145	0.039078	0.008117	0.86488	20	16	3
VIRAT_S_040003_02_000197_000552	5m 54s	1.0281	0.020574	0.068743	0.007356	0.8692	25	21	4
VIRAT_S_050000_05_000696_000732	0m 35s	4.0775	0.089407	0.047252	0.007937	0.73795	3	3	0

6 CONCLUSIONS

We presented a robust tracking framework that uses a probabilistic scheme to combine a motion model of an object with that of its appearance and an estimation of its position. Our tracking framework is robust to abrupt changes in lighting conditions, can follow an object through occlusions. The track starts automatically based on a spatio-temporal algorithm. It can also simultaneously track multiple moving foreground objects of different types (e.g., vehicles, human, etc.) even when they are closely spaced. A "dynamic" integration of the framework with optical flow allows us to track videos resulting from significant camera motion.

We plan to use the results generated by the tracking algorithm to infer trajectory-based events like vehicle turns as well as other complex events like accidents and traffic violations.

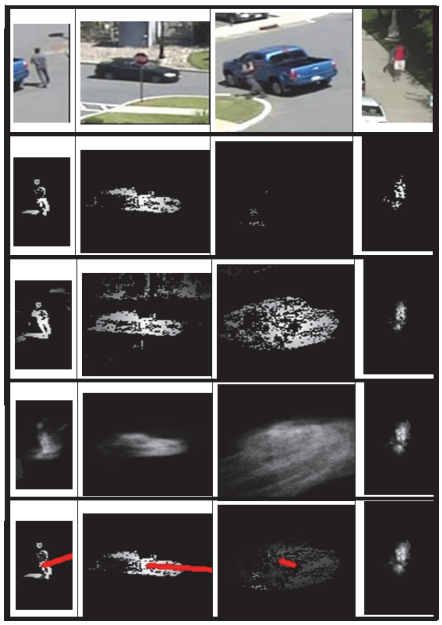

Figure 6: Image of people and cars, the images are the ROI images, followed by MCP, CP, Velocity Image and the Weighted Composite Image from top to bottom.

Table 2: Tracker results for TUD (Andriluka et.al, 2008).

	Campus Correct (False)	Crossing Correct (False)
Expected	303	1008
TUD Detector	227 (0)	692 (7)
MAPTrack	**255 (0)**	**723 (5)**

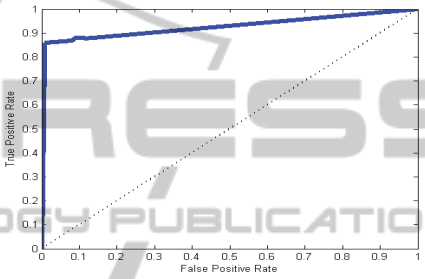

Figure 7: ROC curve for the tracker.

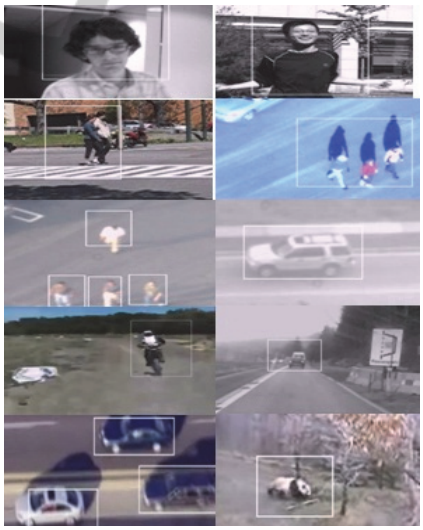

Figure 8: Results from MAPTrack.

ACKNOWLEDGEMENTS

This work has been partially supported by Army Research Office (ARO) under grant number W911NF-10-1-0495. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not

necessarily reflect the views of the ARO or the United States Government.

REFERENCES

Andriluka, M., Roth, S., & Schiele, B. (2008, June). People-tracking-by-detection and people-detection-by-tracking. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on* (pp. 1-8). IEEE.

Babenko, B., Yang, M. H., & Belongie, S. (2009, June). Visual tracking with online multiple instance learning. In Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on (pp. 983-990). IEEE.

Chang, C., & Ansari, R. (2005). Kernel particle filter for visual tracking. Signal processing letters, IEEE, 12(3), 242-245.

Hager, G. D., & Belhumeur, P. N. (1998). Efficient region tracking with parametric models of geometry and illumination. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 20(10), 1025-1039.

Kalal, Z., Matas, J., & Mikolajczyk, K. (2010, June). Pn learning: Bootstrapping binary classifiers by structural constraints. In Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on (pp. 49-56). IEEE.

KaewTraKulPong, P., & Bowden, R. (2002). An improved adaptive background mixture model for real-time tracking with shadow detection. In Video-Based Surveillance Systems (pp. 135-144). Springer US.

Lan, X., & Huttenlocher, D. P. (2004, July). A unified spatio-temporal articulated model for tracking. In Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on (Vol. 1, pp. I-722). IEEE.

Lombardi, P., & Versino, C. (2011). Learning to Detect Event Sequences in Surveillance Streams at Very Low Frame Rate. In Machine Learning for Vision-Based Motion Analysis (pp. 117-144). Springer London.

Lucas, B. D., & Kanade, T. (1981, August). An iterative image registration technique with an application to stereo vision. In IJCAI (Vol. 81, pp. 674-679).

Maggio, E., & Cavallaro, A. (2005, March). Hybrid Particle Filter and Mean Shift tracker with adaptive transition model. In ICASSP (2) (pp. 221-224).

Moreno-Noguer, F., Sanfeliu, A., & Samaras, D. (2008). Dependent multiple cue integration for robust tracking. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 30(4), 670-685.

Oltramari, A., & Lebiere, C. (2012). Using ontologies in a cognitive-grounded system: automatic action recognition in video surveillance. In Proceedings of the 7th international conference on semantic technology for intelligence, defense, and security, Fairfax.

Shi, J., & Tomasi, C. (1994, June). Good features to track. In Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on (pp. 593-600). IEEE.

Smith, K., Gatica-Perez, D., Odobez, J. M., & Ba, S. (2005, June). Evaluating multi-object tracking. In Computer Vision and Pattern Recognition-Workshops, 2005. CVPR Workshops. IEEE Computer Society Conference on (pp. 36-36). IEEE.

Tao, H., Sawhney, H. S., & Kumar, R. (2002). Object tracking with bayesian estimation of dynamic layer representations. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 24(1), 75-89.

Viola, P., & Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. In Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on (Vol. 1, pp. I-511). IEEE.

Virat Public Dataset: http://viratdata.org/

Williams, O., Blake, A., & Cipolla, R. (2003, October). A sparse probabilistic learning algorithm for real-time tracking. In Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on (pp. 353-360). IEEE.