2010 IEEE International Conference on Robotics and Automation
Anchorage Convention District
May 3-8, 2010, Anchorage, Alaska, USA

Visual Tracking and Segmentation using Appearance and Spatial Information of Patches

Junqiu Wang and Yasushi Yagi

Abstract— **Object tracking and segmentation find a wide range of applications in robotics. Tracking and segmentation are difficult in cluttered and dynamic backgrounds. We propose a tracking and segmentation algorithm in which tracking and segmentation are performed consecutively. We separate input images into disjoint patches using an efficient oversegmentation algorithm. Objects and their background are described by bags of patches. We classify the patches in a new frame by searching *k* nearest neighbors. K-d trees are constructed using these patches to reduce computational complexity. Target location is estimated coarsely by running the mean-shift algorithm. Based on the estimated locations, we classify the patches again using appearance and spatial information. This strategy outperforms direct segmentation of patches based on appearance information only. Experimental results show that the proposed algorithm provides good performance on difficult sequences with clutter.**

I. INTRODUCTION

Visual tracking and segmentation in video sequences are essential for action recognition and behavior analysis in robotics. Tracking and segmentation are difficult due to viewpoint variations, occlusions, and dynamic backgrounds. In this work, we use appearance and spatial information in an appropriate way, aiming at partially conquering these difficulties.

One of the crucial problems underlying the construction of reliable tracking and segmentation methods is the identification of characteristic properties of objects [21]. These properties distinguish objects from one another and from the background. Therefore, representation and localization are the main issues to be addressed in designing a robust tracker. We represent targets and their background using bags of patches. We cast tracking and segmentation in two consecutive steps, in contrast with the two most related works [3], [13]. We estimate the location of objects in the tracking process based on appearance information in image patches. In addition, we consider spatial and appearance information for the segmentation process. The objects are segmented by considering appearance and spatial information. In both stages, k-d trees are constructed to accelerate the classification processes. In the tracking stage, we perform approximate localization of targets. This information is then employed in the second stage.

We advocate the use of oversegmentation for object localization and segmentation. Classification at the pixel level is difficult because pixels lack discriminative power, and considering local neighboring information does not alleviate the problem much [3]. Large patches are more discriminative than pixels or small patches. At the same time, however, large patches have less invariant abilities than small patches. Moreover, partitioning an image into large patches tends to produce more errors: objects and background can be grouped into the same patches due to similar appearance. Direct segmentation using an unsupervised algorithm can lead to many errors since these segmentation methods are not always reliable. We must find a tradeoff between invariancy and discriminative power. In this work, we extend graph-based segmentation to a more efficient oversegmentation algorithm. The use of oversegmentation brings a good balance between providing segments that contain enough information for matching and reducing the risk of a segment spanning multiple objects. Oversegmentation also reduces the computational complexity of the algorithm.

Based on oversegmentation results, the tracking process calculates approximate locations for objects. The location information is useful in segmenting an object. Both appearance and spatial information are employed in the segmentation process. We build another k-d tree in which appearance and spatial information are indexed. The image patches are classified again based on the spatial k-d tree. This approach provides better segmentation results than other segmentation methods [13].

This paper is organized as follows. After a brief review of related works in Section II, we introduce the formulation of the consecutive tracking and segmentation in Section III. An oversegmentation algorithm developed from the graph-based segmentation method [9] is described in Section IV. Our tracking approach is introduced in Section V, and the segmentation approach is introduced in Section VI. Section VII describes the experimental results, and conclusions are given in Section VIII.

II. RELATED WORK

Mean-shift tracking [5], [8] is essentially formulated using a non-parametric density gradient estimator to find the image window that is most similar to the object's color histogram in the current frame [17]. Comaniciu et al. [8] define a spatially smooth similarity function and cast the state estimation problem as a search of the basin of attraction of this function. The success of the mean-shift depends strongly on the discriminating power of the histograms that are considered as the object's probability density function. The target representation in the original mean-shift tracker

J. Wang and Y. Yagi are with The Institute of Scientific and Industrial Research, Osaka University, 8-1 Mihogaoka, Ibaraki, Osaka, 567-0047 Japan. Emails: jerywangjq@gmail.com; yagi@am.sanken.osaka-u.ac.jp.

is color histograms. Color histograms are not sufficiently discriminative in many cases. As a global representation of the target, color histograms omit spatial information, which is very important for localization of the target.

Tracking has been considered a two-class separation problem, where a classifier can be trained to distinguish the object from the background. Ensemble tracking [3] trains a classifiers using observations in the reference image. Every pixel is classified using the learned classifier, which results in a confidence map. They run a mean-shift algorithm on confidence maps to localize objects. The classifier is updated based on a least square computation. As mentioned in the previous section, it is difficult to classify pixels into foreground and background using information in pixel-level. Lu and Hager [13] also treat tracking as classification by sampling image patches. Their work achieves good performance on many sequences. However, they do not use spatial information, which is one of the reasons that the confidence maps computed using their approach are not sufficient for challenging sequences. In addition, they partition image into patches using an unsupervised image segmentation algorithm, which can lead the problems detailed in the previous section.

Spatial information has been found useful in tracking. Birchfield and Rangarajan [4] propose a spatiogram-based tracking algorithm to make use of spatial information. They model a target using a histogram in which each bin is spatially weighted by the mean and covariance of the location of the pixels that contribute to that bin. The experimental results in [4] are not satisfying because the spatial information is not well described. Spatial information has also been included in a covariance matrix [16]. Adam et al. [1] build object templates using fixed image fragments. Every patch votes on possible positions of the object in the current frame, by comparing its histogram with the corresponding image patch histogram [15]. This approach can experience difficulties when the objects have deformable structures, which makes the voting process impossible.

Spatial information has been used in our previous work for tracking [23]. In [23], appearance information is described by kernels which are non-parametric; while spatial information is represented by spatial Gaussians. Compared with other works, the tracker [23] provides better likelihood images in some cases. However, it does not aim at segmenting but only tracking. That work decomposes a target into several patches using a parametric description. Such description is not sufficient for complex object segmentation. Moreover, target localization in [23] can be deviated by a cluttered background.

High-level knowledge has been introduced to conquer the problems in tracking [12], [19] using category-level appearance knowledge [18] or dynamics knowledge [20]. Such approaches require reliable a priori object appearance or dynamic models, which must be learned before tracking. Our work aims at dealing with tracking tasks in the absence of an a priori object appearance model. We have little knowledge before tracking and segmentation. The proposed method can handle tracking and segmentation for a wide range of objects, which is important in robotics.

III. Formulation of Our Consecutive Tracking and Segmentation

The objective of this work is to estimate target positions and segmentations from monocular video sequences without any special high-level knowledge. We present a two-stage approach for object tracking and segmentation. In stage 1, moving objects are tracked based on bags-of-patches representation. A bag of patches for each object is initialized by labeling image patches. This stage provides a preliminary estimate of each target's positions and size.

In stage 2, segmentation is formulated as the problem of local partition using appearance and spatial information. We classify image patches again but spatial information is taken into account.

The two stages are interactive in many cases. The segmentation results are occasionally used for the modeling of the objects and background. The segmentation results are better than the approach proposed in [3], [13].

IV. Graph-based Oversegmentation

We represent objects and the background using bags of patches since patches are more discriminative than pixels. To partition an image into patches, we propose an unsupervised oversegmentation algorithm using the graph-based approach [9]. Ideally, the patches should not pass boundaries between different objects or the background.

There are many unsupervised image segmentation algorithms available. Felzenszwalb and Huttenlocher proposed an efficient graph-based image segmentation algorithm. We develop an oversegmentation algorithm based on their work [9] because it is fast in practice and fits into our framework well. We do not use the mean-shift segmentation algorithm [7] because it is less efficient than the graph-based oversegmentation we propose. Mean-shift segmentation finds compact clusters in feature spaces. It generally assumes that the image is piecewise constant, because searching for pixels that are all close together in some feature space implicitly requires that the pixels be alike. Mean-shift finds clusters by dilating each point with a hypersphere of some fixed radius, and then finds connected components of the dilated points. In contrast, the graph-based segmentation for cluster finding does not require all the points in a cluster to lie within any fixed distance since it selects an appropriate dilation radius. In addition, our graph-based oversegmentation is more efficient than mean-shift segmentation.

We build a graph for an image before segmentation. Each vertex in the graph has a corresponding image pixel. Each edge in the graph corresponds to a connection between neighboring pixels. Each pixel is connected to its four neighboring pixels, including the pixels on its top-right, right, bottom, and bottom right. Edge length is computed using the simple Euclidean distance between the color vectors of two pixels.

Fig. 1. The input image on the left is oversegmented into patches. The image on the right show these patches.

To perform single linkage clustering, a minimum spanning tree of the data points is first constructed using Kruskal's algorithm [11]. The edges in the graph are sorted and nodes are merged based on an increasing order of the length of the edges. We define an adaptive predicate for measuring the evidence for a boundary between two regions [9]. Any edges with length greater than a given predicate are removed from the tree. The predicate is adjusted according to the sizes of patches, which is important for the success of the algorithm.

We aim at obtaining oversegmentation of an image using the graph-based algorithm. We calculate an edge length for a precise segmentation for the initialization. A threshold is set according to the edge length. All the edges larger than the threshold are not calculated, which leads to an efficient oversegmentation.

The details of our algorithm are given as follows. Given a graph $G = (V, E)$, we seek an appropriate partition. The number of vertexes, $\|V\|$, is $w \times h$, where w and h are width and height of an image, respectively. The number of edges, $\|E\|$, is roughly $4wh$. We initialize a minimum spanning tree by assigning vertexes in the graph to the leaves in the tree. Each patch contains one pixel (one leaf) after the initialization. Therefore, there are $\|V\|$ patches ($\mathcal{P} = (P_1, \ldots, P_{\|V\|})$) in total. We compute the predicate for two patches (or pixels) connected by an edge. The predicate determines whether the two patches should be merged or not. The predicate depends on the minimum internal difference of a patch, which is defined as the longest edge in the patch.

We sort the edges in an increasing order,

$$E = (e_1, \ldots, e_{\|E\|}),$$

in which $e_i \leq e_j$ if $i < j$. In our oversegmentation algorithm, we give a threshold e_T to stop the merging. All the edges greater than the threshold will not be considered for merging. A constant k is given to control the scale of patches. We set k to 300 in all experiments. In initialization, the minimum internal difference of every patch (pixel) is computed as $\frac{k}{\|P\|}$, where $\|P\|$ is the size of the patch. It is set to k since the size of each patch is 1 in the initialization.

The detailed steps of the proposed oversegmentation algorithm are as follows.

Algorithm: *Graph-based Oversegmentation*

For $e = e_1, \ldots, e_T$,
Assuming e_c is in processing,

1. Let e_c be an edge connecting two vertices v_i and v_j;
2. Let vertex v_i be in a patch $P_{v_i}^{c-1}$ and v_j be in a patch $P_{v_j}^{c-1}$. The patches are the processing results up to the previous edge e_{c-1}. Let e_i^{min} be the minimum internal difference of patch $P_{v_i}^{c-1}$; and e_j^{min} be the minimum internal difference of patch $P_{v_j}^{c-1}$.
3. The two patches $P_{v_i}^{c-1}$ and $P_{v_j}^{c-1}$ are merged if
$$e_c < \min\{e_i^{min}, e_j^{min}\}.$$
4. If the two patches are merged in step 3, the minimum internal difference of the merged patch is updated as
$$e_c + \frac{k}{\|P_{v_j}^{c-1}\| + \|P_{v_i}^{c-1}\|}.$$
Otherwise the patches keep their minimum internal difference unchanged.

End for.

We obtain a set of patches after the iteration without exploring all the edges in the graph, which makes the segmentation more efficient. One oversegmentation example is shown in Fig. 1. We believe that oversegmentation is a good choice for our purpose.

V. Tracking Stage

We run the graph-based oversegmentation algorithm described in the previous section on the first frame. We initialize the tracking by labeling image patches into foreground target and background. We give different labels to different targets when multiple objects are to be tracked.

After the initialization, we represent each target by using two bag of patches. The first bag consists of appearance information, whereas the second bag consists of spatial information. We construct k-d trees to accelerate the searching of the nearest neighbors. The classification is carried out through nearest neighbor searching in the k-d trees. We

classify image patches into foreground and background based on the proportion of foreground and background patches in the nearest neighbors. We also use backward checking techniques which are useful in improving classification results. We run the mean-shift algorithm on the confidence map produced by the classification. The rough position of a target is estimated. The rough position will be used in the next section for segmentation.

A. K-D Tree Construction

The computational cost of nearest-neighbor searching is rather expensive, which includes finding the neighbors and storing the entire training set [10]. With N observations and p predictors, nearest-neighbor classification requires Np operations to find the neighbors per query point.

K-d trees use a space partitioning data structure for saving input data into k-dimensional space, which results in an efficient search involving a multidimensional search key. K-d trees are a special case of Binary Space Partitioning (BSP) trees. They organize data using a specific rule to choose axis-aligned splitting planes. The structure is helpful for reducing time complexity of k-nearest neighbor searching.

We use the efficient approximate nearest neighbor algorithm and k-d tree algorithm in [14]. We build k-d trees both in the tracking and segmentation stages. In the tracking stage, the input data is 3D. The expected time for a nearest neighbor search is $O(\log(N))$, where N is the number of elements stored in the k-d trees. k-d trees require no learning or training of parameters. The preprocessing step has low complexity in the number of elements and has a short runtime.

B. Confidence maps

We build a k-d tree using 3-tuples (R,G,B). Each object is represented using a bag of patches. The background is also represented using a bag of patches. Only appearance information is employed in this stage. We classify image patches by searching nearest neighbors in the k-d tree.

We partition the new frame into patches using the proposed oversegmentation algorithm. These patches need to be classified into foreground or background. We perform forward and backward checking for the classification. In the forward checking, we build a k-d tree for all the patches that represent targets and background. For each image patch in the frame, we search for their k_F^T nearest neighbors in the k-d tree (The subscript T denotes *Tracking* and the superscript F denotes *Forward checking*). We set k_F^T to 12 in all the experiments. Assuming that m_F^T patches in the k_F^T nearest neighbors belong to the foreground model, we assign the foreground confidence value c_F^T with

$$c_F^T = m_F^T. \tag{1}$$

In the backward checking, we build a k-d tree for the patches in the current frame. We find k_B^T nearest neighbors for all the patches in the foreground model. We set k_B^T to 12. Assuming one patch in the current frame is the n_B^Tth nearest neighbor of a foreground patch, we compute the foreground confidence value c_B^T of this patch as

$$c_B^T = k_B^T - n_B^T. \tag{2}$$

We compute the final confidence value c^T of a patch in tracking stage by

$$c^T = \frac{c_F^T + c_B^T}{k_F^T + k_B^T}. \tag{3}$$

VI. Segmentation State

We construct another k-d tree using (R, G, B, X, Y) 5-tuples for a frame to be classified. We use the object model with spatial information to search for nearest neighbors in the spatial k-d tree. Note that the last two elements in the tuples are adjusted to the current frame according to the rough position estimate.

Similar to the formulation in tracking stage, we carry out forward checking and backward checking based on nearest neighbor searching results. In the forward checking, we build a k-d tree for the target and background models. We find nearest neighbors for each patch in current frame. The confidence value c_F^S in forward checking is computed as $c_F^S = m_F^S$, where m_F^S is the number of foreground model patches in the nearest neighbor searching results.

In the backward checking, we build a k-d tree for the patches in current frame. We search nearest neighbors for each patch in the target model. The foreground confidence value c_B^S in backward checking is computed as $c_B^S = k_B^S - n_B^S$, where k_B^S is the nearest neighbor setting, and n_B^S is the rank of a patch in target model.

The final confidence value in segmentation stage is computed using

$$c^S = \frac{c_F^S + c_B^S}{k_F^S + k_B^S}. \tag{4}$$

The classification result in segmentation stage is much better than that in tracking stage since both appearance and spatial information are considered for classification.

A. *Target Re-localization*

The confidence map produced in this stage is better than that in the tracking stage since spatial information is employed in the classification. We can estimate better bounding boxes of targets by running the mean-shift algorithm again. The re-localization can give a better location of the target [5].

B. *Computational Complexity Analysis*

Compared with classification using exhaustive searching, the proposed tracking algorithm has lower computational costs. We analyze the computational complexity to show the advantage of our algorithm.

Assuming the number of patches to be matched in an image is N; the number of patches in a target and background model is M, we compare the computational costs in tracking and segmentation stages with and without the k-d tree searching. In the tracking stage, Exhaustive search for forward and backward matching involves $O(2MN)$ times

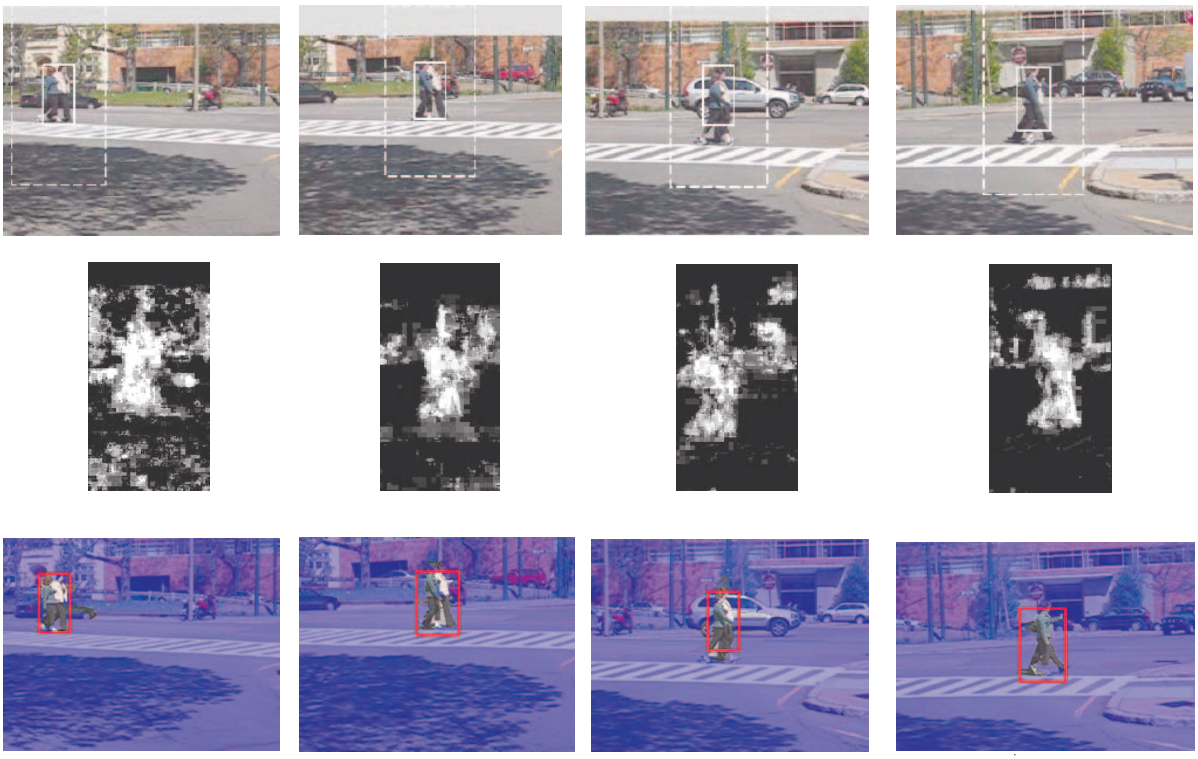

Fig. 2. Segmentation results of a street sequence. The images in the first row are the tracking results of the ensemble tracking [3]. The images in the second row are the confidence maps for each frame in the first row. The confidence maps correspond to the dashed rectangle. The images in the third row are the tracking and segmentation results of the proposed algorithm.

of vector distance calculation. It takes $O(N\log(M)) + O(M\log(N))$ times calculation when we build k-d trees for the matching. In the segmentation stage, exhaustive searching for forward and backward matching involves $O(2MN)$ times of vector distance calculation. Using k-d trees, $O(N\log(M)) + O(M\log(N))$ times matching for the vectors is needed for the segmentation. Therefore, the complexity ratio between exhaustive matching and k-d tree based searching is approximately $\frac{O(2MN))}{O(N\log(M))+O(M\log(N)}$. Considering the cost for building the k-d trees, the ratio is $\frac{O(2MN)}{O(N\log(M))+O(M\log(N))+O(N\log(N))+O(M\log(M))}$. N is always much larger than M. Note that we use the over-segmentation algorithm, thus the patch number tends to be large. In all the test sequences, patch matching using k-d trees is much more efficient than exhaustive matching. The advantage of the k-d trees is more apparent when foreground targets have large sizes.

VII. Experimental Results

We have implemented the proposed tracking and segmentation algorithm. It was tested on many challenging image sequences. The testing results of two sequences are shown here. While some segmentation algorithms [13] have achieved success in image sequences with little additional clutter, they have not been successfully applied to the kind of cluttered images that we consider in this work.

The first sequence (the street sequence) was captured by a hand-held camera. The tracking results are shown in Fig. 2. We compare the proposed approach with the ensemble tracker [3]. The ensemble tracker [3] essentially is a binary classifier learning online. The objective of that tracker is to classify pixels into foreground and background. The localization is performed based on the classification results. The background in Fig. 2 changes abruptly from time to time; as a result, the ensemble tracker [3] does not produce reliable confidence maps. The confidence maps it produced are shown in the second row of Fig. 2. Note that these confidence maps are post-processing results of pixel classification by discarding ambiguous foreground pixels. The ensemble tracking algorithm can track the pedestrians successfully through the sequence. However, there are many segmentation errors in the results. The proposed consecutive tracking and segmentation algorithm classifies the patches iteratively and gets better segmentation results, as shown in the third row of Fig. 2.

The second sequence is from the public Carnegie Mellon University (CMU) dataset with ground truth [6]. The car is occluded by trees in the sequence, which brings difficulty to tracking and segmentation. As shown in Fig. 3, the car is tracked and segmented successfully in the frames having occlusions.

These experiments demonstrate that our tracking and segmentation algorithm is useful for object tracking and segmentation in cluttered background. Spatial information is one of the reasons that good results were obtained in the experiments.

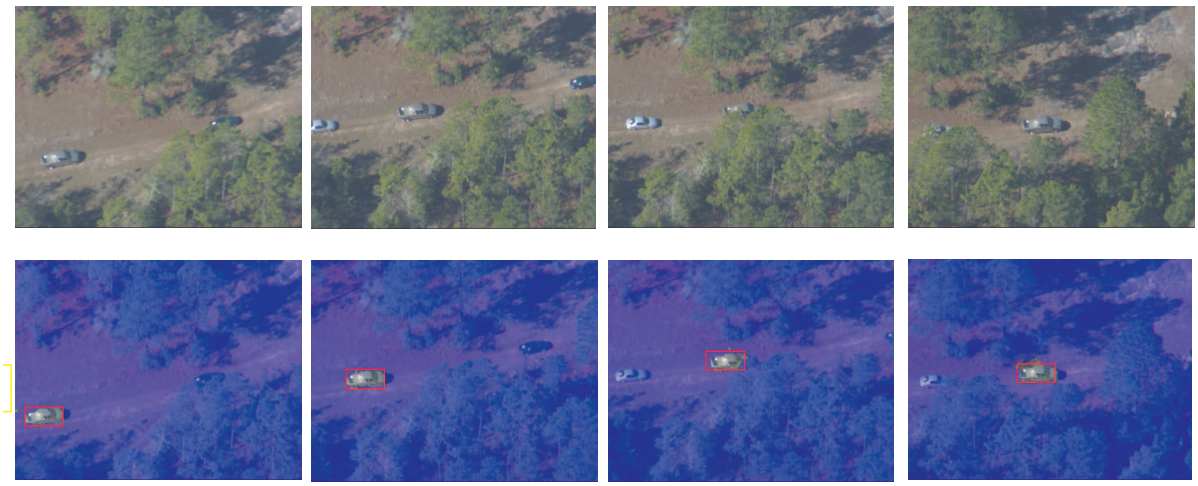

Fig. 3. Segmentation results of a street sequence from the CMU dataset. The images in the first row are the input frames and the images in the second row show the tracking and segmentation results of the proposed algorithm.

VIII. Conclusions and Future Work

We propose an effective consecutive tracking and segmentation algorithm. Tracking using appearance information is conducted by classifying image patches. The estimated location information is combined with appearance and spatial information to separate objects from background. We build k-d trees for accelerating the tracking and segmentation.

Although the proposed oversegmentation algorithm is efficient and helpful in improving segmentation performance, it cannot avoid errors due to the limitation of the bottom-up segmentation approach. We are interested in further improving segmentation performance using shape information in the future.

We adopt the mean-shift algorithm to do localization on confidence maps. The mean-shift algorithm requires the object kernels in the consecutive frames to have a certain overlap. This requirement cannot be met when the motion between consecutive frames is large. We are going to use the strategies presented in [24] to deal with this problem.

References

[1] A. Adam, E. Rivlin and I. Shimshoni. Robust fragments-based tracking using the integral histograms, in *Proc. of Computer Vision and Pattern Recognition 2006*.

[2] S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for on-line non-linear/non-gaussian bayesian tracking, *IEEE Trans. on Signal Processing*, 50(2): 174-188, 2002.

[3] S. Avidan. Ensemble tracking, *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(2): 261–271, (2007).

[4] S. Birchfield and S. Rangarajan. Spatiograms versus histograms for region-based tracking, in *Proc. of Computer Vision and Pattern Recognition 2005*.

[5] G. Bradski. Computer vision face tracking as a component of a perceptual user interface, in *Proc. of the IEEE Workshop Applications of Computer Vision 1998*.

[6] R. Collins, X. Zhou, and S. Teh. An open source tracking testbed and evaluation web site, in *Proc. of IEEE Int'l Workshop on Performance Evaluation of Tracking and Surveillance 2005 (PETS 2005)*.

[7] D. Comaniciu and P. Meer. Mean shift: a robust approach toward feature space analysis, *IEEE Trans. Pattern Anal. Mach. Intell.* 24(5): 603–619 (2002).

[8] D. Comaniciu, V. Ramesh, and P. Meer. Kernel-based object tracking. *IEEE Trans. Pattern Anal. Mach. Intell.*, 25(5): 564–577 (2003).

[9] P. Felzenszwalb and D. Huttenlocher. Efficient graph-based image segmentation, *Int'l Journal of Computer Vision*, 61(1): 55–79 (2005).

[10] J.H. Friedman, J. Bentley, and R.A. Finkel. An algorithm for finding best matches in logarithmic expected time, *ACM Trans. Math. Softw.*, 3(3): 209-226 (1977).

[11] J. Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society*, 7(1): 48–50 (1956).

[12] Y. Li, H. Ai, T. Yamashita, S. Lao, and M. Kawade. Tracking in low frame rate video: a cascade particle filter with discriminative observers of different life spans, in *Proc. of Computer Vision and Pattern Recognition 2007*.

[13] L. Lu and G.D. Hager. A nonparametric treatment for location/ segmentation based visual tracking, in *Proc. of Computer Vision and Pattern Recognition 2007*.

[14] D. Mount and S. Arya. Ann: A libary for approximate nearest neighbor searching, in *Proc. of CGC 2nd Annual Workshop on Comp. Geometry 1997*.

[15] F. Porikli. "Integral histogram: a fast way to extract histograms in cartesian spaces," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, Vol. 1, pp. 829–836, June 2005.

[16] F. Porikli, O. Tuzel and P. Meer. Covariance Tracking using Model Update Based on Lie Algebra, in *Proc. of Computer Vision and Pattern Recognition 2006*.

[17] M. Swain and D. Ballard, "Color Indexing," *Int'l. Journal of Computer Vision*, 7(1): 11–32, 1991.

[18] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features, in *Proc. of Computer Vision and Pattern Recognition 2001*.

[19] J. Wang and Y. Yagi. Consecutive tracking and segmentation using adaptive mean-shift and graph cuts, in *Proc. of IEEE Int. Conf. on Robotics and Biomimetics 2007*.

[20] J. Wang, Y. Makihara, and Y. Yagi. Human tracking and segmentation supported by Silhouette-based Gait Recognition, in *Proc. of IEEE Int'l. Conf. on Robotics and Automation 2008*.

[21] J.Y.A. Wang and E.H. Adelson. Representing moving images with layers, *IEEE Trans. on Image Processing*, 3(5): 625–638, (1994).

[22] J. Wang, Y. Yagi. Integrating Color and Shape-texture Features for Adaptive Real-time Tracking, *IEEE Trans. on Image Processing*, 17(2): 235–240, (2008).

[23] J. Wang, Y. Yagi. Patch-based adaptive tracking using spatial and appearance information, in *Proc. of IEEE Int'l. Conf. on Image Processing 2008*, pp. 1564-1567.

[24] J. Wang, Y. Yagi. Adaptive mean-shift tracking with auxiliary particles, *IEEE Trans. on Systems, Man and Cybernetics -Part B*, vol. 39(6), pp.1578-1589, 2009.

[25] M. Yang, J. Yuan and Y. Wu. Spatial Selection for Attentional Visual Tracking, in *Proc. of Computer Vision and Pattern Recognition*, 2007.