On-the-fly Object Modeling while Tracking

Zhaozheng Yin and Robert Collins
Department of Computer Science and Engineering
The Pennsylvania State University, University Park, PA 16802
{zyin,rcollins}@cse.psu.edu

Abstract

To implement a persistent tracker, we build a set of view-dependent object appearance models adaptively and automatically while tracking an object under different viewing angles. This collection of acquired models is indexed with respect to the view sphere. The acquired models aid recovery from tracking failure due to occlusion and changing view angle. In this paper, view-dependent object appearance is represented by intensity patches around detected Harris corners. The intensity patches from a model are matched to the current frame by solving a bipartite linear assignment problem with outlier exclusion and missed inlier recovery. Based on these reliable matches, the change in object rotation, translation and scale is estimated between consecutive frames using Procrustes analysis. The experimental results show good performance using a collection of view-specific patch-based models for detection and tracking of vehicles in low-resolution airborne video.

1. Introduction

To achieve long-term object tracking, a persistent tracker must adapt to changes in object and background appearance while avoiding drift during the adaptation [9]. Furthermore, since object trackers suffer from losing the object with high probability during occlusion and changes in view angle, a persistent tracker must be able to reacquire the object after these failures [5]. Our approach to solve these problems is to model the object adaptively and automatically while tracking, so that the object can be detected and recognized again after losing it (Fig.1). The models, which are collections of small view-dependent intensity patches, are accumulated on-the-fly during the tracking run.

Learning object models for later recognition remains a challenging problem in the fields of machine learning and computer vision. Several recent approaches have been based on the observation that features belonging to the same object should have correlated behavior (e.g. should be seen together frequently or should move in the same direction) while features belonging to different objects exhibit more independent or uncorrelated behavior [4, 14, 16]. Previous approaches are good at discovering view-dependent cliques of features that have consistent behavior, but they leave open the problem of inferring chains of cliques that represent different views of the same object (e.g. the two sides and front view of the same vehicle).

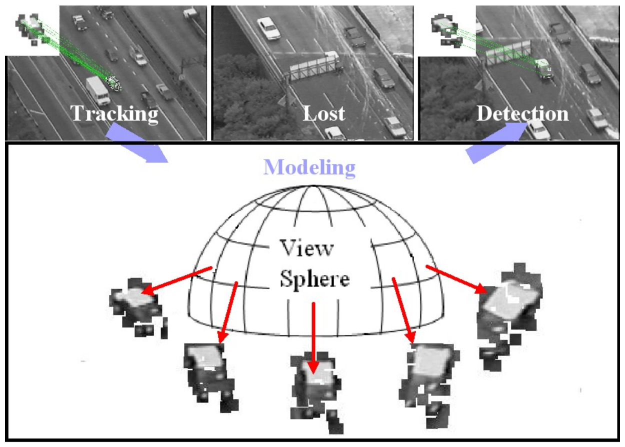

Figure 1. A collection of view-dependent intensity patch models built while tracking can help the persistent tracker recover from failure by enabling global detection of the target object.

Rothganger et al. introduced a successful 3D object representation using local affine-invariant image descriptors constrained by spatial relations between the corresponding surface patches [15]. The invariants select promising local patches for modeling, and the spatial constraints efficiently match the same object patches under different views by discarding geometrically inconsistent candidate matches. In practice, this approach relies on textured objects and high resolution images, and the 3D object models are constructed in controlled environments with little or no clutter. Thus, it is not feasible to build models for less textured objects in cluttered low resolution video, e.g. airborne video of vehicles in traffic. Dowson and Bowden provided a simultaneous modeling and tracking (SMAT) method [2]. The object features are selected manually and tracked individually. While tracking, a mixture model is fit to the exemplar appearances and geometric structures built from the feature positions. This approach is closely related to the template updating work of [9]. Leordeanu et al. developed an algorithm for unsupervised learning of object models as constellations of features [7]. Small clusters of features that match across frames in short subsequences are grouped together to form "parts". The novelty of this approach is that transitive chains of part pairs that span different aspects of the same object can be discovered.

Our approach builds an object model during real-time tracking. We represent the object by small intensity patches around detected Harris corners [6]. The patches from the previous model are matched to the current frame by solving a bipartite linear assignment problem. Furthermore, outlier matches are excluded by an Exhaustive Sample Consensus process, and some missing inlier matches are recovered using geometric location constraints. Based on correct matches found, the elliptical shape change in object rotation, translation and scale between two consecutive frames is estimated using Procrustes analysis. The above processes, including corner detection, patch extraction and matching, and Procrustes analysis, run in real-time to provide continual modeling while tracking. While tracking the object under different view angles, we collect and index view-dependent object patches with respect to a viewing sphere. The result is an appearance-based model that explicitly characterizes change in object appearance with respect to pose, and that can be used to recognize the same object at a later time from an unknown (but previously seen) viewpoint.

Each acquired appearance-based object model consists of a view sphere populated with viewpoint dependent object representations (e.g. clusters of intensity patches). This provides a valuable resource for recognition because it completely characterizes change in object appearance with respect to viewing angle. Note that the view sphere will not necessarily be fully populated from a single video sequence - it may take several passes of the sensor to see all aspects of the object. However, the view-sphere appearance model succinctly categorizes which views have been seen, and which have not yet been seen, and thus can be used proactively to suggest new viewpoints for modeling previously unseen sides of the object.

In the rest of this paper, we describe simultaneous object modeling and tracking in Section 2. Indexing object models using a view sphere is introduced in Section 3 and experimental results are shown in Section 4.

2. Object Modeling while Tracking

Instead of tracking an object by warping and matching a single large template, in this approach we represent the object by a constellation of intensity patches. The patch features are extracted around Harris corners. The object is tracked in new images by matching the patches using normalized cross correlation (NCC). Since an individual intensity patch does not include the object's rotation and scale change information between consecutive frames, we apply Procrustes analysis on the locations of all well-matched object patches to estimate the similarity transformation.

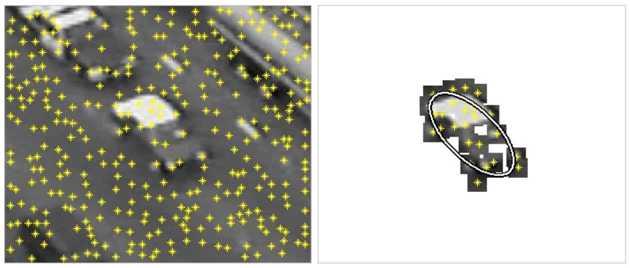

Figure 2. Left: detected Harris corners within a subimage; Right: the object is constrained by an elliptical shape and represented by intensity patches extracted around Harris corners.

2.1. Feature detector and descriptor

Good feature detectors and descriptors are important for object modeling and tracking. Quite a few state-of-the-art feature detectors and descriptors have been evaluated based on their performance on sets of test images under different image conditions [10, 11, 12]. Generally, the combination of Harris/Hessian-affine detectors and SIFT feature descriptor [8] performs the best. In the Harris-affine feature detection method, the Harris corner detector determines interest points and a Laplacian of Gaussian filter is used to select the appropriate scale. To achieve invariance to affine transformations, the second moment matrix of the intensity gradients is used to estimate the shape of a covariant elliptical region that is used to normalize the image neighborhood.

In this paper, we also use the Harris corner detector to find interest points, but represent local appearance using intensity patches centered at each Harris point (Fig.2). In the first input image, the target is chosen manually and constrained by an elliptical shape. The Harris corners within or close to the ellipse belong to the object. Instead of using the whole object appearance as template, the object is represented by a set of intensity patches that are extracted around Harris corners. Although the whole appearance of the object is likely to change within consecutive frames, some of the local patches between two consecutive frames may remain approximately the same. Thus we achieve feature similarity from these small patches without iteratively warping the whole appearance template for matching.

In our approach, a Harris corner detector with non-max suppression is used to locate interest points. Image appearance features are extracted around Harris corners and represented by 11×11 intensity patches. Each patch has several additional properties: patch location in the image, mean and variance of the pixel intensities within the patch, and feature ID. With normalization by subtracting the mean and dividing by the standard deviation, the patches become invariant to global illumination changes. There are several reasons to choose the 11×11 window size. First, we need to maintain the feature descriptors in a low dimensional space without losing distinctiveness. If the patch is too large, it requires more computational and memory cost. Additionally, if a square patch is large enough to cover the whole object and the object changes its appearance between two consecutive frames, it is hard to find a perfect matching between these two frames. If the patch is too small (the extreme case is a one-pixel patch), many outlier matches could be found between two frames, thus there is low distinctiveness. Secondly, an 11×11 square patch can be converted and padded into a 128-byte vector for efficient computation. For example, the normalized cross correlation between patches can be implemented efficiently by dot product between two normalized vectors using MMX instructions [13].

2.2. Feature matching

During tracking, we use the object patch model built from previous frames to match the object in the current input image. After detecting Harris corners and extracting corresponding intensity patches in the current image, we need to determine which matches between object model patches and current image patches are acceptable. We consider this as a bipartite linear assignment problem (LAP) or "marriage problem". Every object patch is compared via NCC (implemented by MMX vector dot product) with all input image patches. The one having the highest correlation score is considered as a mate of this object patch. Similarly, every input image patch is correlated with all object patches, with the highest scoring one yielding a mate. If the two mate pairings agree, they are a good match. Fig.3 illustrates a set of matches found by solving the marriage problem. Many outlier matches exist because of nearby similar objects (clutter).

Random Sample Consensus (RANSAC) is usually used to exclude outliers from a large data set. Since our objects are small, there are not many matching patches in our video sequences, so we can perform Exhaustive Sample Consensus (EXSAC) to select the best inlier set. The difference between EXSAC and RANSAC is that every combination of k points (e.g. 3 for affine transformation) is tried when finding inliers, not just a random sampling. Fig.4 gives the result after an affine EXSAC is used to exclude outliers. We assume the previous patch model and object in the current image is related by an affine transformation:

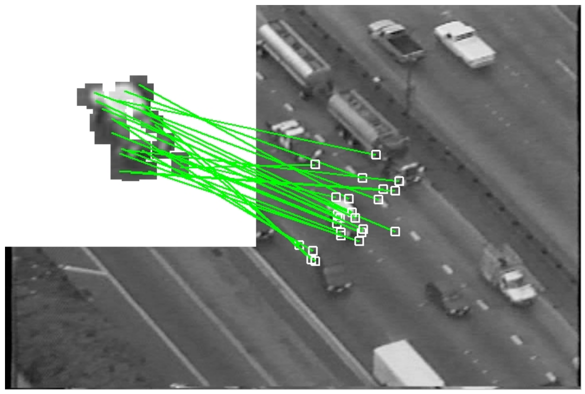
Figure 3. Initial matches are found by solving a bipartite linear assignment problem.

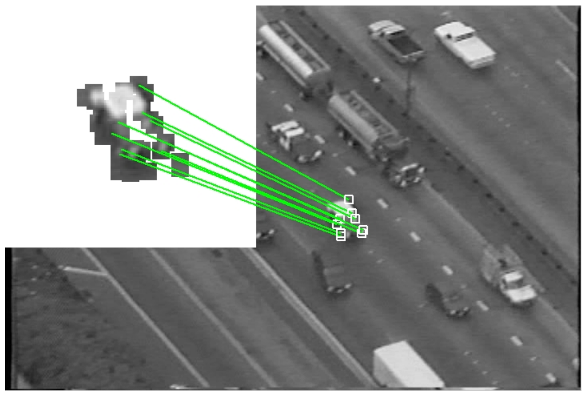
Figure 4. Inlier matches found by exhaustive sample consensus.

$$\begin{pmatrix} x_i' \\ y_i' \\ 1 \end{pmatrix} = \begin{pmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix} \qquad (1)$$

where $(x_i' \ y_i' \ 1)$ is the matched patch location in the current image and $(x_i \ y_i \ 1)$ is the matched patch location in the model. A sum of weighted square errors is minimized to estimate the affine transform parameters $h = (h_1, h_2, h_3, h_4, h_5, h_6)$:

$$\min_h \sum_i \omega_i \{(h_1 x_i + h_2 y_i + h_3 - x_i')^2 + (h_4 x_i + h_5 y_i + h_6 - y_i')^2\} \qquad (2)$$

The weight ω_i for each matched pair is proportional to their matching score and life length of their feature ID (how many times the feature ID appeared in the past L frames). As experiments show, most of the features do not last long and we can choose a small history window (e.g. $L = 5$ or 10). We use the weight to avoid drifting in cluttered backgrounds, i.e. matches with higher matching score that exist longer are more reliable than others. This assumes

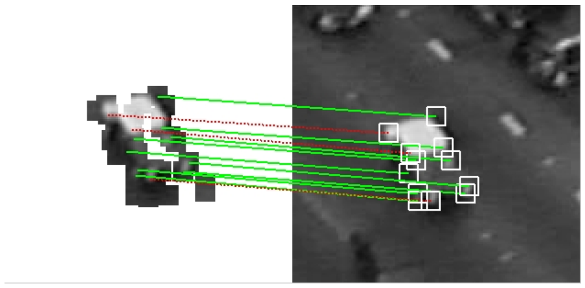

Figure 5. Recovering some previously missed inlier matches (red).

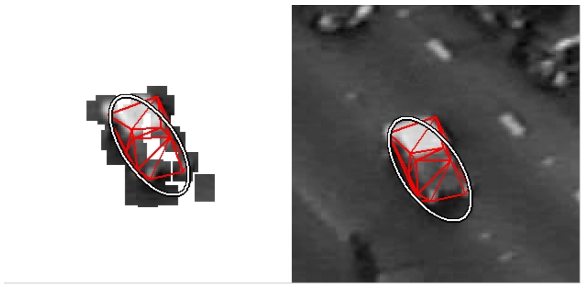

Figure 6. Shape matching between previous and current Delaunay graphs to estimate the similarity transformation of the object's bounding ellipse.

that features on the moving object persist within the tracking window, whereas features in the background leave the window fairly quickly. Modeling the background area surrounding the foreground target or adaptively classifying the foreground from the background [1] could also be explored to avoid drifting. The solution to the optimization problem (Eq.2) is:

$$(h_1\ h_2\ h_3)^T = A^{-1}b_1 \quad (h_4\ h_5\ h_6)^T = A^{-1}b_2 \tag{3}$$

where

$$A = \sum \begin{pmatrix} \omega_i x_i^2 & \omega_i x_i y_i & \omega_i x_i \\ \omega_i x_i y_i & \omega_i y_i^2 & \omega_i y_i \\ \omega_i x_i & \omega_i y_i & \omega_i \end{pmatrix}$$

and

$$b_1 = \sum \left(\omega_i x_i^{'} x_i,\ \omega_i x_i^{'} y_i,\ \omega_i x_i^{'}\right)^T$$
$$b_2 = \sum \left(\omega_i y_i^{'} x_i,\ \omega_i y_i^{'} y_i,\ \omega_i y_i^{'}\right)^T$$

The above matching process is executed between the object appearance patch model and the whole sub-image. The bidirectional matching is very strict and spatial information of the patches is not taken into account. However, after we estimate the affine transformation, we can recover some missed inlier matches by restricting the searching range and matching these features again. Specifically, the location of one unmatched model patch can be affine transformed into the current image coordinates, and we search for this model patch's mate around just that position instead of the whole subimage. If some image patch is close to the model patch and their matching score is above a threshold, we consider this as a good match. As a result of this process, we can recover some of the inlier matches that were missed previously. Fig.5 shows the result after recovery of missed inliers.

After finding matches by solving bipartite LAP, excluding outliers by affine EXSAC and recovering missed inliers, we have a set of reliable matches. The feature ID of each image patch is determined in two cases: if a current image patch is matched to a patch in the model, it inherits the feature ID from its matched model patch; otherwise this is a new image patch and a new feature ID is assigned to it.

2.3. Procrustes Analysis on Matched Patch Locations

The locations of the matched patches form a Delaunay Triangulation graph, which is a useful global shape constraint. The object appearance patch model can be enhanced by this geometric structure. Locally, intensity patches around Harris corners describe the object appearance details, but they are blind to their neighboring patches. Globally, the Delaunay Triangulation connects all the object patches into a network graph. Although the object orientation and scale change between consecutive frames can not be estimated from a single pair of intensity patch matches, we can rely on the spatial Delaunay graph to compute the similarity transformation of the object's bounding ellipse between consecutive frames. We model object shape (ellipse) changes using 4-parameter similarity rather than 6-parameter affine to increase robustness, so the elliptical shape can only translate, rotate and scale in an image.

We apply Procrustes analysis ([3]) on the two Delaunay graphs of consecutive frames to compute the similarity transform between them (translation (t_x, t_y), rotation θ and scale s). The position, orientation and scale of the current ellipse is computed based on the similarity transformation and previous ellipse properties: position, orientation, semimajor and semiminor axis lengths. The intensity patches around the new ellipse are considered as a new model for tracking the object in the next frame. Fig.6 illustrates the Procrustes analysis result between two consecutive frames.

3. Indexing Object Models using a View Sphere

For small changes of viewpoint, such as between consecutive frames, object features can be matched well and thus the object can be tracked in the video by chaining these short subsequences, as shown in Fig.7. Both SIFT keys and

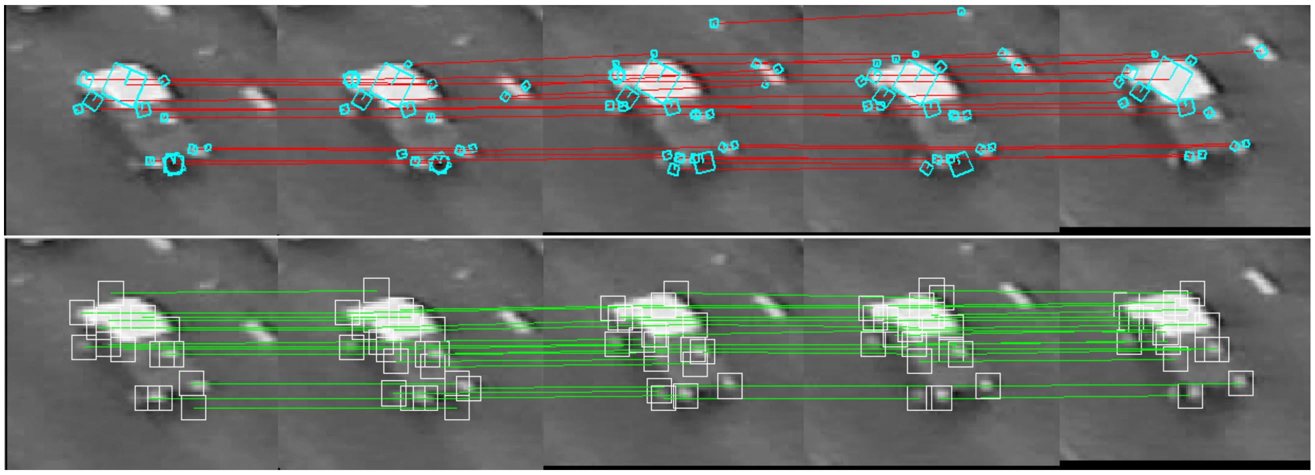
Figure 7. Top row: SIFT key matches within five consecutive frames; Bottom row: corner-patch matches within five consecutive frames.

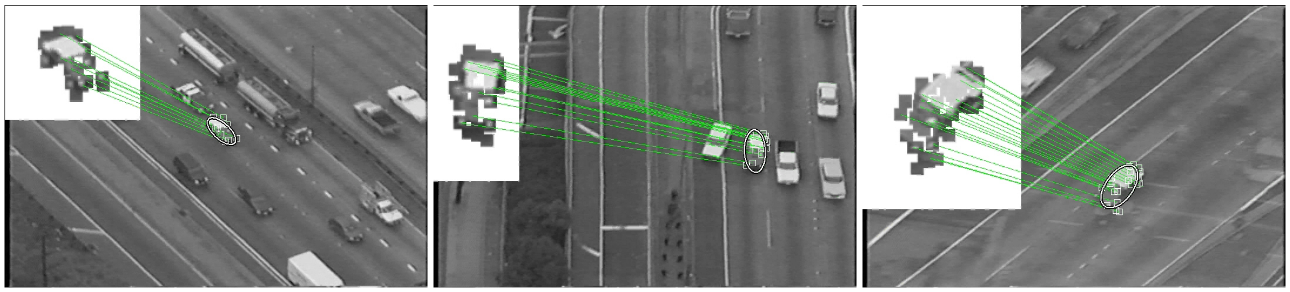
Figure 8. A sample training sequence for acquiring view-dependent object models. The camera circles around the vehicle as it travels along a straight highway.

our corner patches can be matched well between consecutive frames, but our approach is faster and more features are extracted.

However, recent comparison has shown that the repeatability performance of state-of-the-art detectors degrades slowly with increasing change of viewpoint [11], and no detector-descriptor combination performs well with view point changes of more than 25-30 degrees [12]. This inspired us to build a collection of view-specific object models arranged on a view sphere, as described below.

3.1. Collecting Patch Models

Fig.8 shows a sample training sequence used to build a patch model adaptively and automatically while tracking. Fig.9 shows the feature numbers at each time instant and corresponding ellipse orientations. There are about 15-40 matched features at each time instant (when the object size increases in the image, more features are extracted and matched). In this sequence, the vehicle travels along a straight highway while the airborne camera circles around it. The orientation of object bounding ellipse in the image thus evolves simultaneously with change in viewing angle.

Fig.10 shows the status of each feature, where each column represents life time of one feature and the rows show when this feature appeared during the tracking. This view matrix shows that most features do not last long in the sequences, i.e. building a single model with viewpoint invariance would be quite difficult, thus it is necessary to track the object by chaining together object models acquired under different view angles.

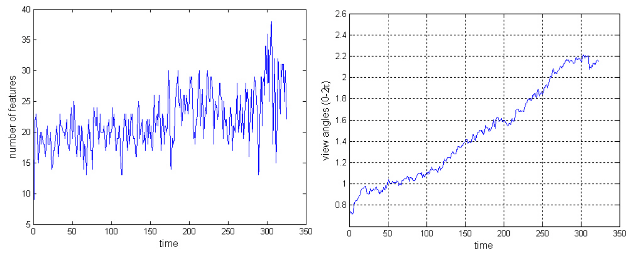

Figure 9. Left column: the total number of features at each time instant; Right column: the ellipse orientation at each time instant.

3.2. Indexing by Viewing Angle

The viewing angle of a camera can be specified by two angles, θ and ϕ, which correspond respectively to the az-

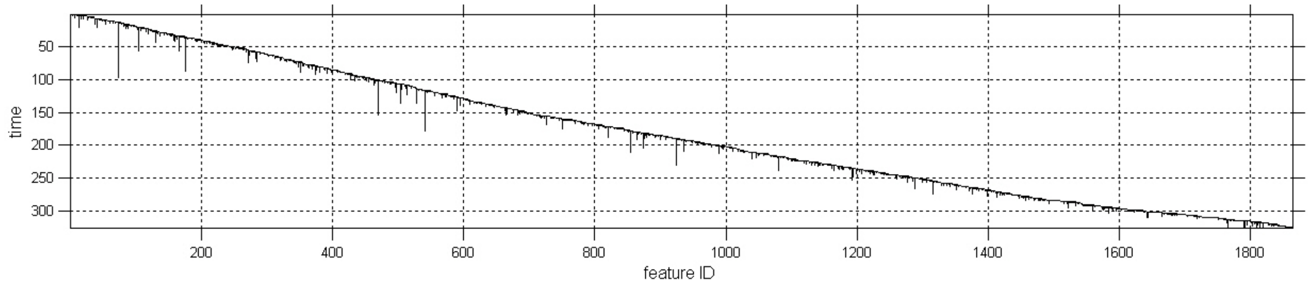

Figure 10. The birth and death process of each feature ID.

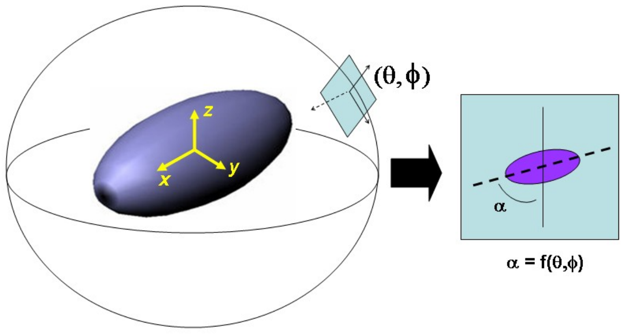

Figure 11. View angle computation.

imuth and elevation of the principal viewing ray with respect to a view sphere centered on the object. In our current application, we use a heuristic to compute azimuth angle of the view from the orientation of the major axis of the bounding ellipse of a vehicle in the image. This heuristic assumes a view with large elevation angles (say roughly 70-90 degrees) such as those taken by a circling airborne camera.

Assume the vehicle can be bounded with a 3D prolate (cigar-shaped) ellipsoid, with the long axis aligned with an object-centered X-axis (Fig.11). A camera with viewing angle (θ, ϕ) with respect to a viewing sphere centered at the origin of this ellipsoid has relative rotation

$$R = \begin{bmatrix} -\sin\theta & \cos\theta & 0 \\ \sin\phi\cos\theta & \sin\phi\sin\theta & -\cos\phi \\ -\cos\phi\cos\theta & -\cos\phi\sin\theta & -\sin\phi \end{bmatrix}$$

Assuming the camera is distant enough from the object that orthographic projection is relevant, the relationship between 3D object-centered coordinates X,Y,Z and image coordinates x,y is specified as

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} -\sin\theta & \cos\theta & 0 \\ \sin\phi\cos\theta & \sin\phi\sin\theta & -\cos\phi \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

Under this projection model, the major axis of the ellipsoid, namely X,Y,Z = 1,0,0 maps to a vector $(-\sin\theta, \sin\phi\cos\theta)$ with respect to the 2D center of mass of the object in the image. Although this vector contains both elevation and azimuth angle information, it is not possible to tease them apart without further information, and thus not possible to uniquely determine viewing angle from just the bounding ellipse in the image. However, for large enough elevation angles, we can approximate $\sin\phi \approx 1$, such that the orientation of the major axis of the bounding ellipse in the image is directly related to just the azimuth angle of the camera view. Azimuth angle can thus be approximately recovered by applying arc tangent to the components of the major axis of the 2D bounding ellipse of the vehicle in the image. In our modeling experiments, we cluster acquired models by this recovered angle to form a discrete set of model clusters indexed by approximate azimuth angle.

Instead of retaining an unsorted bag of patches, we sample and index the patch models by view angle on the view sphere (Fig.1). When detecting the object globally in an image to recover from tracker failure, "lookup" of the nearest model on the sphere is more efficient than exhaustive search through an unsorted bag of models for one with similar appearance. Our viewing angle computation is simplified and based on eccentricity here because we are looking at nearly textureless objects (cars) in low-resolution video. In close-range cases with textured objects, more sophisticated view angle estimates can be computed via SFM [15].

4. Experimental results

An object tracker based on SIFT keys and graph matching has been successfully implemented by Tang and Tao [17], which takes on average 3fps (P4 3.2GHz, C++). Compared to their SIFT key tracker, our algorithm runs at 15 fps on average (P4 3.2GHz, Matlab). If the object is larger, it takes more time to detect and extract the corner patches and find reliable matches.

Fig.8 demonstrates a scene in which the camera circles around a moving vehicle. Fig.12 shows another tracking sequence where the object turns around a corner, yielding a quick change in object appearance. In addition, rapid panning motion of the camera occasionally makes the image blurred. For comparison, we also tested two template matching methods based on Lucas-Kanade template warping: (1) updating the template adaptively frame by frame and (2) always matching with the initial selected template. They both failed during this sequence. Fig.13 shows an example where the object is static and the camera moves around it. Both illumination and object appearance change during the tracking. In all these sequences, the objects are tracked and modeled well by intensity patches extracted around detected Harris corners.

After we build the object patch models on the view sphere, they can be used for detection if the target is lost in new images. For example, the azimuth angle changes about 90 degrees in the training sequence of Fig.8, and we sample this view angle space into 20 collections of object patches and use them for detection in new video sequences. Fig.14 shows some of the detection results using this set of models, when the object is partly occluded or seen from slightly different viewpoints. The detection is performed by solving a bipartite LAP with outlier exclusion and missed inlier recovery, as described in Section 2.2. Each collection of object patches searches for its match over the entire new image frame, therefore the detection takes more time (0.3 seconds per patch model on a 480×720 image) than the tracking process, in which only the subimage around the object is searched. Fig.14 shows examples of using patches learned from one sequences to detect the same object in another new sequence by recognizing it in previously seen poses, and multiple collections of object patches can be matched to the target in the test image because those patches are similar to each other. On the other hand, this duplicate detection by multiple patch models increases the detection accuracy.

5. Conclusion

To persistently track an object by its appearance, the model must adapt to appearance change and avoid drifting. Since existing feature detectors and descriptors can not perform well under large changes in view angle, we construct a collection of view-specific appearance models on a view sphere. The object is represented by a constellation of intensity patches extracted around detected Harris corners. The patch model is built automatically while tracking the object by solving a bipartite LAP patch matching using outlier exclusion and missed inlier recovery. The scale and rotation change of the object is estimated by applying Procrustes analysis on a spatial graph of patch locations. The experimental results show that the algorithm performs well when tracking objects, and that the view-specific models acquired can be used to detect the object again in new frames if the tracker has lost it. Since no detector and descriptor can outperform all others under all image conditions [11], in future work we will consider combining complementary detectors and descriptors together to enhance the ability to construct appearance models while tracking.

Acknowledgment

This work was funded under the NSF Computer Vision program via grant IIS-0535324 on Persistent Tracking.

References

[1] S. Avidan. Ensemble Tracking. IEEE Trans. Pattern Anal. and Machine Intell., 29(2): 261-271, February 2007.

[2] N. Dowson and R. Bowden. Simultaneous modeling and tracking (SMAT) of feature sets. CVPR2005, pp.99-105.

[3] I. Dryden and K. Mardia. Statistical Shape Analysis. John Wiley, Chichester, 1998.

[4] R.Fergus, P.Perona, and A. Zisserman. Object class recognition by unsupervised scale-invariant learning. CVPR2003, pp.264-271.

[5] Y. Guo et.al. Vehicle Fingerprinting for Reacquisition and Tracking in Videos. CVPR2005, pp.761-768.

[6] C. Harris and M. Stephens. A Combined Corner and Edge Detector. Fourth Alvey Vision Conference, 1988.

[7] M. Leordeanu, R. Collins and M. Hebert. Unsupervised Learning of Object Features from Video Sequences. CVPR2005, pp.1142-1149.

[8] D. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. International Journal of Computer Vision (IJCV), 60(2):91-110, 2004.

[9] I.Matthews, T.Iashikawa and S.Baker. The Template Update Problem. IEEE Trans. Pattern Anal. and Machine Intell., 26(6):810-815, June 2004.

[10] K. Mikolajczyk and C. Schmid. A Performance Evaluation of Local Descriptor. IEEE Trans. Pattern Anal. and Machine Intell., 27(10):1615-1630, October 2005.

[11] K. Mikolajczyk et.al. A Comparison of Affine Region Detectors. IJCV, 65(1):43-72, 2005.

[12] P. Moreels and P. Perona. Evaluation of Feature detectors and Descriptors based on 3D Objects. IJCV, 73(3):263-284, 2007.

[13] D. Nister, O. Naroditsky and J. Bergen. Visual Odometry. CVPR2004, pp.652-659.

[14] D. Ramanan and D. Forsyth. Using Temporal Coherence to Build Models of Animals. ICCV2003, pp.338-345.

[15] F. Rothganger et.al. 3D Object Modeling and Recognition Using Local Affine-Invariant Image Descriptors and Multi-View Spatial COnstraints. IJCV, 66(3):231-259, 2006.

[16] J. Sivic and A. Zisserman. Video Data Mining Using Configurations of Viewpoint Invariant Regions. CVPR2004, pp.488-495.

[17] F. Tang, and H. Tao. Object tracking with dynamic feature graph. IEEE Workshop on VS-PETS, 2005, pp.25-32.

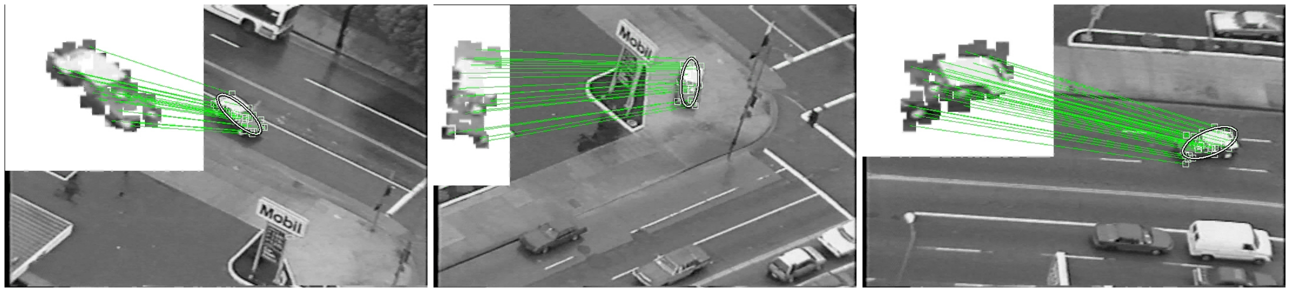

Figure 12. The object turns around a corner, yielding a quick change in object appearance.

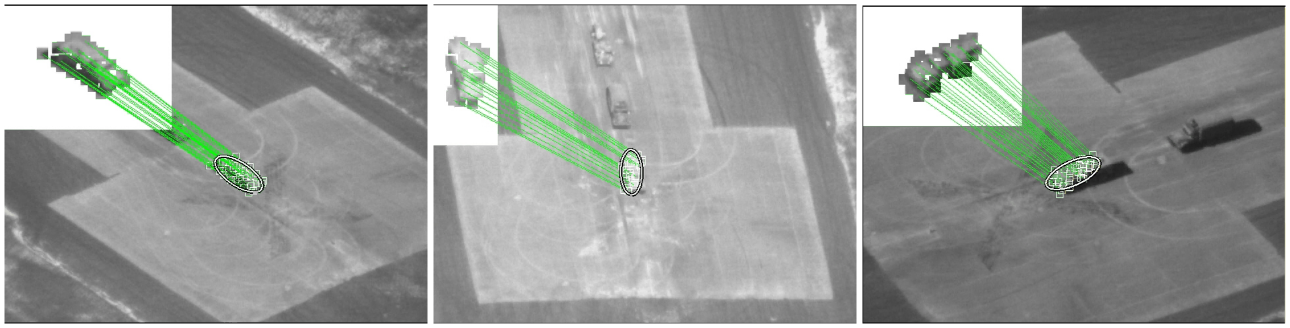

Figure 13. Large changes in viewpoint as a camera moves around a static truck.

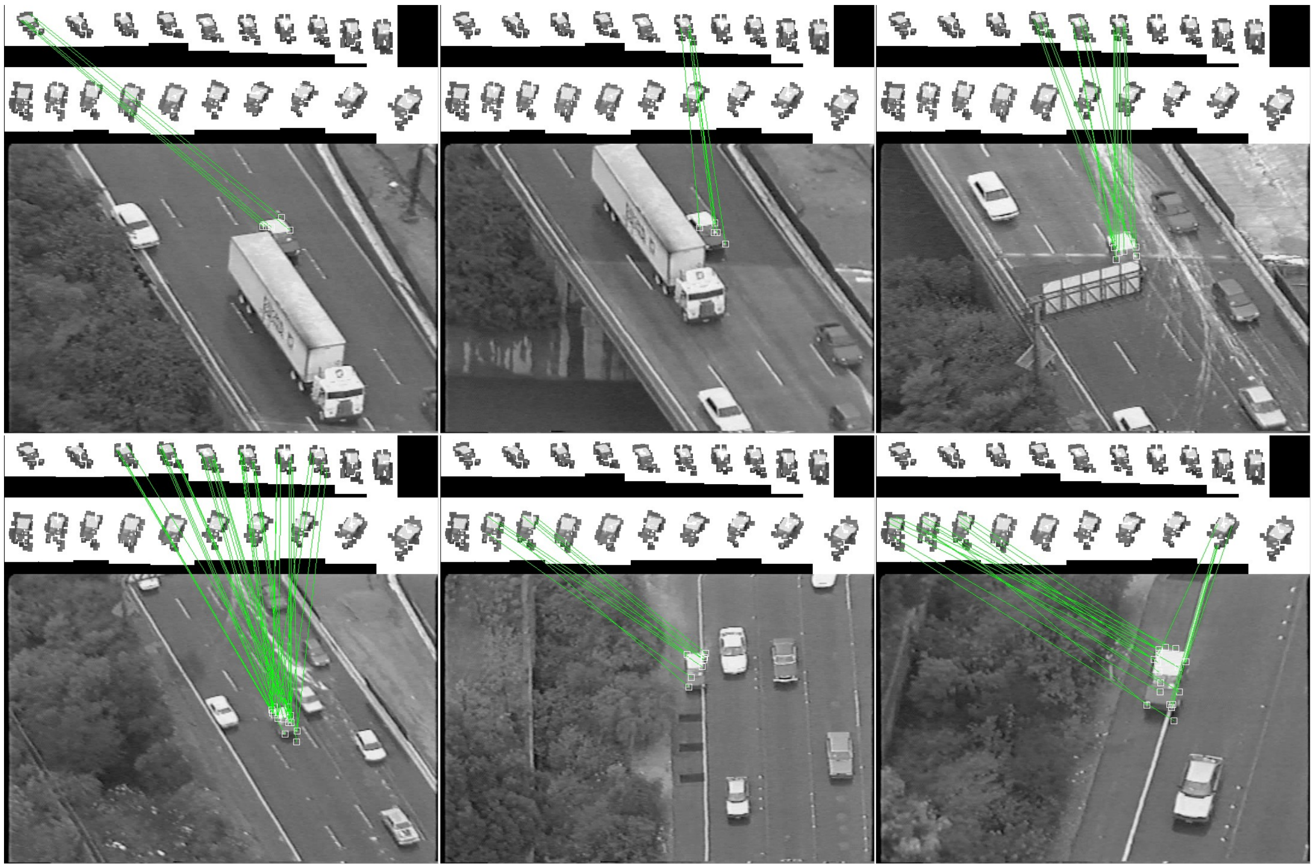

Figure 14. Detecting the object under occlusions and different view angles using a collection of models learned previously.