

Non-Rigid Object Localization and Segmentation Using Eigenspace Representation

Omar Arif and Patricio Antonio Vela
School of Electrical and Computer Engineering
Georgia Institute of Technology, Atlanta, GA 30332
omararif@gatech.edu, pvela@ece.gatech.edu

Abstract

This paper presents a novel non-rigid object localization and segmentation algorithm using an eigenspace representation. Previous approaches to eigenspace methods for object tracking use vectorized image regions as observations, whereas the proposed method uses each individual pixel as an observation. Localization using the pixel-wise eigenspace representation is robust to noise and occlusions. A unique feature of the approach is that it permits segmentation in addition to localization. Localization and segmentation are carried out by deriving a similarity function in the eigenspace. The algorithm is tested on synthetic and real world tracking examples to demonstrate the performance.

1. Introduction

We focus on visual tracking of deformable objects in this paper. The motion of a deformable object can be divided into an overall global motion (pose), and the local deformation of the object [19]. Tracking a deformable object is therefore, the process of estimating the pose parameters (*Target Localization*) and the local deformation (*Target Segmentation*) of the object.

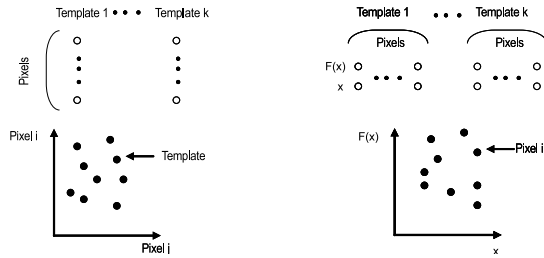
Many tracking methods have been proposed ranging from local feature-based to global template-based. Local feature based methods track key points in the image where significant changes occur, such as, corners, edges, location with significant texture changes [9, 10, 14]. Feature-based methods [12] typically require a minimum size to ensure a consistent, trackable feature-density for the targets. Alternatively, features can be extracted through a statistical analysis of the track target data [15].

Global-template-based trackers determine the correspondence of the object region in consecutive images by using a template or a set of templates of the target object to define a region descriptor, which provides characteristic information about the object. Tracking is then formulated

as comparing region descriptors. For example, Comaniciu et al [6] use kernelized histograms as a probability density function of the object region. To find the most probable target position in the current frame, Bhattacharyya coefficient is used as a dissimilarity measure between a target distribution (color distribution) and a candidate region distribution. Position is efficiently computed using mean-shift iterations. Other template-based methods learn the appearance of the object using supervised or unsupervised machine learning techniques. Avidan [2] constructs a feature vector for every pixel in the reference image and trains several weak binary classifiers to separate pixels that belong to the object from the pixels that belong to the background. A single strong classifier is tested on all pixels in the current image to create a confidence map. Mean shift is run on the confidence map to find the object rectangle. Unsupervised learning techniques such as principal component analysis (**PCA**) can be used to represent the templates in a low dimensional eigenspace. The eigenspace representation provides a compact notion of the target being tracked rather than treating the target as a set of independent pixels. This representation has been used for tracking in [3] and [11]. In [11], the subspace is also incrementally updated to account for appearance and illumination change. Chin et al [5] use nonlinear subspace representation derived using kernel methods [16] for face recognition and visual tracking.

In the eigenspace methods mentioned above, the templates are vectorized to form a matrix $\mathbb{D} = [I_1, \dots, I_N]$, where each column I_i is a vectorized template. Dimension reduction methods (i.e., PCA or kernel PCA) are used to obtain the eigenspaces. In this setting each template I_i as an observation with the implicit assumption that the image appearance will remain similar to the training templates. However, under partial occlusions this assumption will not hold and the tracker may give erroneous results.

Contribution. This paper proposes a way to incorporate appearance plus additional target information into a nonlinear eigenspace representation. The method is schematically described in Figure 1(b). Instead of considering a vector-



(a) Each template is an observation. (b) Proposed approach: Each pixel Training templates are vectorized, stacked together, and learned. Pixel vector (appearance + spatial location) is an observation. Pixel vectors from all training templates are amassed and learned.

Figure 1. Target representations: template-wise vs. pixel-wise.

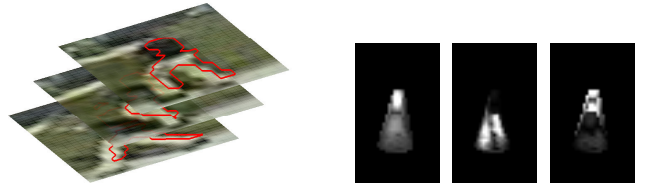
ized target template as an observation, each pixel vector of the target template is an observation, where a pixel vector may include color value, spatial location of the pixel, gradient, edge value etc. Pixel vectors extracted from a single template can be used for learning the model. Such an approach is not feasible for other eigenspace methods, where several to many vectorized object templates are required. The representation is powerful in the sense that tracking is done pixel-wise, but the pixels are tied together in eigenspace representation. The quadratic similarity measure used during tracking operates in a probabilistic manner: pixels that match the model score high, whereas pixels that do not score low. This characteristic provides robustness to occlusions. An additional feature of the representation is that it also generates segmentations, thus extending eigenspace tracking algorithms to include identification of the object silhouette. Lastly, a variational method is provided for more efficient target localization.

2. The Target Model

The image \mathcal{I} is represented as a two-dimensional lattice of a one-dimensional intensity image or a three-dimensional color image. Let $\mathcal{F}(x)$ be the p -dimensional appearance vector extracted from \mathcal{I} at the spatial location x ,

$$\mathcal{F}(x) = \Gamma(\mathcal{I}, x), \quad (1)$$

where Γ can be any mapping such as color, image gradient, edge, texture etc. The lattice domain is the *spatial* domain, while the p dimensional appearance attributes from the *appearance* domain. A pixel vector is constructed by concatenating the appearance and the spatial values in the joint appearance-spatial domain of dimension $d = p + 2$. Let $u = [\mathcal{F}(x), x]^T$ be such a d dimensional pixel vector, representing a pixel at location x in the joint appearance-spatial domain [8]. To build the target model, one or more template images containing the target are selected (typically, from the first few frames of the video sequence). A



(a) Extracting appearance and spatial information from a set of segmented templates. One template image will suffice for localization only. Multiple templates are required for segmentation.

(b) Absolute values of the first three eigenvectors. The eigenvectors divide the object into different parts, which is helpful during occlusions where only the eigenvectors corresponding to visible parts contribute to localization.

Figure 2. Target model.

consistent track point within each template is chosen (e.g., the waist of a person such as in Figure 2(a)) and presumed to be the origin of the spatial domain. The templates are then manually segmented to extract the pixel vectors. The set of all pixel vectors define the target input space \mathbb{D} ,

$$\mathbb{D} = \{u_1, u_2, \dots, u_n\}. \quad (2)$$

where n is the total number of pixel vectors extracted from the template images. If only localization is desired, then one template image will suffice. To also segment the target, pixel vectors from multiple templates are required for learning the feasible shape deformations.

2.1. KPCA-based Eigenspace Representation

The eigenspace representation of the input space comes from kernel PCA (KPCA) [16], which maps the data into a high-dimensional space using a nonlinear mapping $\phi : \mathbb{R}^d \rightarrow \mathcal{H}$ and utilizes the covariance matrix in that space. In [15] the covariance matrix in the input space is used for tracking. The covariance matrix extracts second order linear dependencies. KPCA captures nonlinear statistical relations among the pixels, such as the relationship among three or more pixels in an edge or a curve [13]. Furthermore, PCA can only capture a number of features equal to the dimension of the input space, thus its representation capacity is limited. KPCA can represent a number of features equal to the number of eigenvalues retained plus one [7], thus its representation capacity is richer.

Elgammal [8] employs a joint appearance-spatial space in a probabilistic setting and uses the Kullback-Leibler information distance as a similarity measure between two image regions. However, it is known that Bhattacharya coefficient or Kullback-Leibler based similarity functions may not provide sufficient discrimination when the true distribution is sparsely allocated within a high-dimensional feature space (appearance + spatial) [22]. Also, these techniques require sophisticated space partitioning and/or correction

strategies [18].

The KPCA eigenvectors are obtained by diagonalizing the kernel matrix K given by $K_{ij} = \mathbf{k}(u_i, u_j)$, where \mathbf{k} is a kernel such as the Gaussian kernel,

$$\mathbf{k}(u_i, u_j) = \exp\left(-\frac{1}{2}(u_i - u_j)^T \Sigma^{-1}(u_i - u_j)\right), \quad (3)$$

with Σ is a $d \times d$ diagonal matrix of bandwidths for each appearance-spatial coordinate, $\{\sigma_{F_1}, \dots, \sigma_{F_p}, \sigma_{s_1}, \sigma_{s_2}\}$. A test point x is represented in the KPCA space by projecting it onto the eigenvectors. The projection of the k^{th} eigenvector is

$$f_k(u) = \sum_{i=1}^n \frac{\alpha_i^k}{\sqrt{\lambda^k}} \mathbf{k}(u_i, u). \quad (4)$$

where $\alpha^k = [\alpha_1^k, \dots, \alpha_n^k]^T$ and λ^k are the k^{th} eigenvector and eigenvalue of the kernel matrix K . Figure 2(b) shows absolute values of the eigenvectors learned using KPCA.

The projection Equation (4) is computed in the high-dimensional space through the kernel in terms of linear combinations of all the input vectors $\{u_i\}_{i=1}^n$. If the total number of elements n is large, (4) becomes unsuitable for on line tracking applications. The projection equation can be approximated using fewer samples $c_i^k \in \mathbb{R}^d$ with coefficients w_i^k as

$$f_k^*(u) = \sum_{i=1}^l w_i^k \mathbf{k}(c_i, u). \quad (5)$$

The points c_i^k and their coefficients w_i^k are found using a reduced kernel representation (see [1] and references therein).

3. Object Tracking

Object tracking in the eigenspace pixel representation, will require a similarity function in this same eigenspace. That similarity function will measure the similarity of a pixel vector observation to the learned model. As per §2.1, the similarity function defined in the eigenspace will capture nonlinear relationships between the pixel vectors.

3.1. Pixel Vector Similarity Function

The KPCA space is a high-dimensional elliptical space with the pixel vectors in the target model \mathbb{D} living on the surface of the ellipse, as is evident from the function $f_k(u)$, which tends to zero as the vector u recedes from the input space \mathbb{D} . Pixel vectors far from the target input space \mathbb{D} will map to the vicinity of the origin in KPCA space. Also since $\|\alpha^k\|^2 = 1$ and $\langle \phi(u_i), \phi(u_j) \rangle \leq \mathbf{k}(u_i, u_j) \leq 1$, there is an upper bound for $|f_k|$, $|f_k| \leq \sqrt{\lambda^k}$, where λ^k is the k^{th} eigenvalue. The net result is to map the collection of data into a high-dimensional elliptical space. An element that

coincides or agrees with the data will map onto the surface of the hyper ellipse, whereas one that does not will map close to the origin based on the degree to which it is an outlier. Hence the squared distance from the origin operates like similarity measure [20], with the further property that it is functionally similar to a likelihood. The similarity of a test pixel vector u to the target model \mathbb{D} being

$$\mathcal{J}(u) = \sum_{k=1}^m (f_k(u))^2. \quad (6)$$

where m is the total number of eigenvectors retained.

3.2. Region similarity measure

To measure the similarity of a region \mathcal{R} , all pixel vectors u falling within the region \mathcal{R} are extracted. The center of the region is taken to be the origin of the spatial domain. Using the similarity measure $\mathcal{J}(\cdot)$ (Equation (6)), computes the similarity of u to the target model, \mathbb{D} . Repeating the calculation for all the vectors in the region \mathcal{R} then taking the sum results in a measure of how close, as a whole, the region \mathcal{R} represent the target model. The measure of region similarity, arising from the k^{th} eigenvector is

$$\mathcal{J}_k(\mathcal{R}) = \sum_{u \in \mathcal{R}} \mathcal{J}_k(u) = \sum_{u \in \mathcal{R}} (f_k(u))^2 \quad (7)$$

For multiple eigenvectors, sum the contributions of each eigenvector to get the complete region similarity measure. The center location of the region \mathcal{R} with the maximum $\mathcal{J}(\mathcal{R})$ will most likely correspond to the location of the target. Figure 3 shows the computation of region similarity measure for the first three eigenvectors and the sum of the region similarity measures for these eigenvectors. In all the cases, the peak is found at the location of the target. This means that the target can be robustly located even under partial occlusions, where the eigenvectors corresponding to visible parts are used to track the object.

3.3. Variational Target Localization

Assume that the target object undergoes a geometric transformation to a region $\tilde{\mathcal{R}}$, such that $\mathcal{R} = T(\tilde{\mathcal{R}}, a)$, where a is the transformation parameter. Let \hat{u} be a pixel vector sampled from the region $\tilde{\mathcal{R}}$ whereby $\hat{u} = [\mathcal{I}(\tilde{x}), T(\tilde{x}, a)]^T = [\mathcal{I}(\tilde{x}), x]^T$. This section derives a gradient descent procedure to maximize the region similarity (Equation (7)) with respect to the transformation parameter a . The gradient is

$$\nabla_a \mathcal{J}_k(\tilde{\mathcal{R}}) = \sum_{\hat{u} \in \tilde{\mathcal{R}}} \nabla_a \mathcal{J}_k(\hat{u}) = \sum_{\hat{u} \in \tilde{\mathcal{R}}} 2f_k(\hat{u}) \nabla_a f_k(\hat{u}), \quad (8)$$

where

$$\nabla_a f_k(\hat{u}) = \nabla_a T(\tilde{x}, a) \cdot \nabla_x f_k(\hat{u}), \quad (9)$$

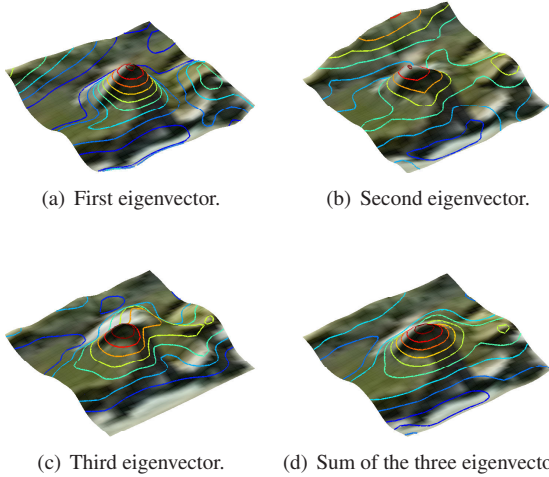


Figure 3. Region similarity measure for the first three eigenvectors. For each eigenvector, the region similarity peaks at the location of the target object. A target can be robustly located even under occlusions, as the eigenvectors corresponding to visible portions will score higher and thus have more influence.

with $\nabla_a T(\tilde{x}, a)$ a $g \times 2$ Jacobian matrix of T given by $\nabla_a T = [\frac{\partial T}{\partial a_1}, \dots, \frac{\partial T}{\partial a_g}]^T$, where g is the number of transformation parameters. The gradient $\nabla_x f_k(\hat{u})$ is

$$\nabla_x f_k(\hat{u}) = \frac{1}{\sigma_s^2} \sum_{i=1}^l w_i^k \mathbf{k}(c_i^k, \hat{u}, \cdot) (\pi_s(c_i^k) - x), \quad (10)$$

where π_s is a function that takes full d -dimensional vector and returns only the spatial values, and σ_s is the spatial bandwidth parameter of the kernel \mathbf{k} . The transformation parameters are then updated using the following equation:

$$a(t+1) = a(t) + \delta t \sum_{k=1}^m \nabla_a \mathcal{J}_k(\tilde{\mathcal{R}}) \quad (11)$$

where m is the total number of eigenvectors used and δt is the time step.

An alternate update using mean-shift: When only translational motion is considered, the mean shift method [4] can be used to find the location of the target object. Assume \hat{x} is the estimated location of the target object, and the region $\mathcal{R}_{\hat{x}}$ is centered at \hat{x} . Let the feature vector u be given by $u = [\mathcal{I}(x), x - \hat{x}]$. Due to the probabilistic properties of the similarity measure (§3.1), $\mathcal{J}(\mathcal{R}_{\hat{x}})$ represents an unnormalized density estimate computed at \hat{x} . To find the local mode mean shift iterations can be used. Mean shift is a non-parametric, iterative procedure for locating stationary points of a density function. The update location is

$$\hat{x} = \frac{\sum_{u \in \mathcal{R}_{\hat{x}}} \sum_{k=1}^m f_k(u) \sum_{i=1}^l w_i^k \mathbf{k}(c_i^k, u) (\pi_s(c_i^k) - x)}{\mathcal{J}(\mathcal{R}_{\hat{x}})}. \quad (12)$$

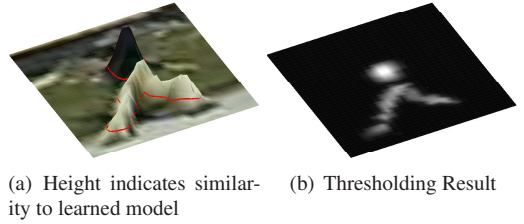


Figure 4. Segmentation by thresholding

Iterating (12) finds the region whose similarity measure is the local density maximum.

3.4. Segmentation by Thresholding

Once the target has been localized, to perform segmentation the similarity measure (Equation (6)) is computed for each pixel vector falling within the region \mathcal{R} and thresholded to get a binary mask for the target object. In Figure 4(a), the image that falls within the region \mathcal{R} is used to color the surface which is created as a result of measuring the similarity of each pixel vector to the learned model. The contour in the figure represents the thresholding of the values to get the binary mask shown in Figure 4(b).

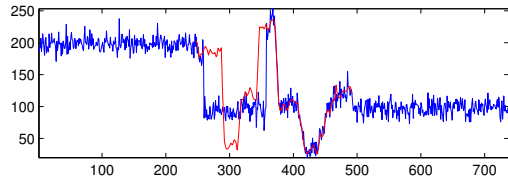
4. Experiments

4.1. 1D Synthetic Signal Detection

First we test the eigenspace representation and the corresponding region similarity measure (Equation (7)) on a 1D synthetic example as in [17]. Consider detecting a 1D template signal embedded in a random signal of thrice its length and corrupted by (1) Gaussian noise, (2) log-normal noise or (3) occlusions. The corrupted signal is searched over the whole domain to find the template. Comparison algorithms include (1) kernel density correlation (**KDC**) [17], (2) covariance-based detection (**COV**) [15], and (3) sample correlation (**SC**). KDC, based on kernel density estimation, is used for stereo registration and tracking [17]. KDC outperforms Mutual Information, a standard measure used for robust estimation [21].

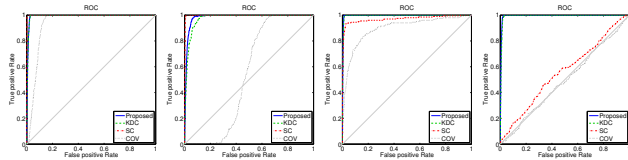
Figure 5 depicts an instance of the random signal, in which the template signal is embedded starting at 247 with 40% occlusion and Gaussian noise. The original template signal is shown in red. For all experiments, the true location of the template signal in the random signal is fixed at 247. The number of eigenvectors m used to compute the region similarity is 8. To build the vectors, the signal value and its spatial location is used. For each set of parameters (noise level / % occlusion), the experiment is repeated 200 times, and the location of the embedded signal is estimated.

Detection performance is shown by plotting the detec-



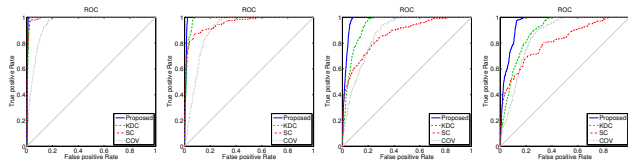
(a) Blue: Corrupted signal with Gaussian noise and 40% occlusion, Red: Original template signal.

Figure 5. Template signal and a sample corrupted signal.



(a) Gaussian, (b) Gaussian, (c) Log-Normal, (d) Log-Normal, $\mu = 0, \sigma = 30$. $\mu = 0, \sigma = 60$. $\mu = 0, \sigma = 2$. $\mu = 0, \sigma = 4$.

Figure 6. ROC curves for Gaussian and Log Normal noise.



(a) 10%. (b) 30%. (c) 50%. (d) 60%.

Figure 7. ROC curve for different levels of occlusion.

tion probability (true positives) vs. the false alarm probability, as the discrimination threshold is varied. The plot, known as the receiver operative characteristic (ROC), is widely used to analyze detector performance. The closer the curve hugs the upper-left corner, the better the performance. In Experiment 1, the signal is corrupted at different i.i.d Gaussian noise levels. ROC curves for noise levels 30 and 60 are shown in 6(a) and 6(b). SC performs the best; under additive white Gaussian noise, SC is the minimum variance unbiased estimator. Next, log-normal noise is added to simulate background clutter. Results are depicted in Figures 6(c) and 6(d). SC is sensitive to outliers, so its performance deteriorates. The proposed measure and KDC are robust to outliers. Finally, performance under occlusion is tested. The template signal is occluded at random locations for occlusions of 10% to 60%. The results, shown in Figure 7, clearly show the proposed measure’s robustness to occlusions. The results show that the proposed method has much better ROC curves for noise and for occlusions. The same is expected for higher dimensional source data.

4.2. Object Localization and Segmentation

The tracking performance of the proposed method is tested on a number of real world objects such as cars, a dog, a fish, walking and running people, and face tracking.

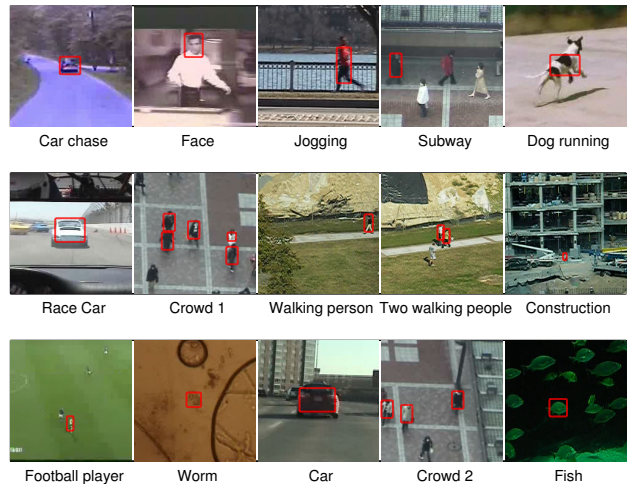
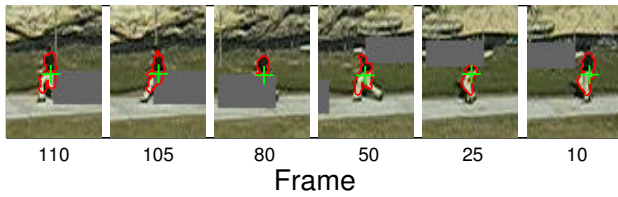


Figure 8. Tracked Objects

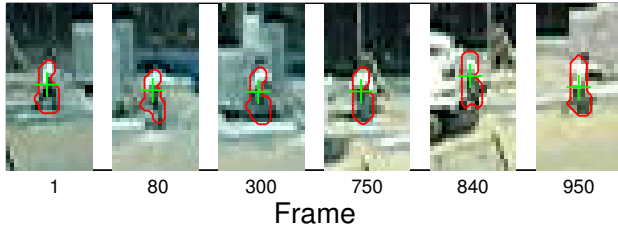
Some videos are standard test sequences with scene clutter, noise, scale changes, and occlusions. In some, the target is small compared to the image dimensions. To build pixel vectors, the color values (RGB) and the spatial values were used. The value of σ in the Gaussian kernel for learning the target model is $\sigma_F = 50$ for the color values and $\sigma_s = 5$ for the spatial domain. The number of eigenvectors used are in the range $m = [3, 5]$. The learned space is reduced using [1]. As a result Equation (5) was used for computing the projections with $l = [8, 12]$. The proposed tracker was implemented in MATLAB on a Intel Core2 1.86 GHz processor with 2GB RAM. Run-time for the proposed tracker is about 1 frame/sec for all experiments.

Target Localization: We first consider localizing the objects in Figure 8. Only translation motion is considered and the update Equation (12) is used. Segmentation is not performed and only one template image is used. The tracker successfully tracks all the objects except for the walking person and jogging sequence. In these sequences, the object undergoes complete occlusion at which point the tracker, because of its variational nature, loses track.

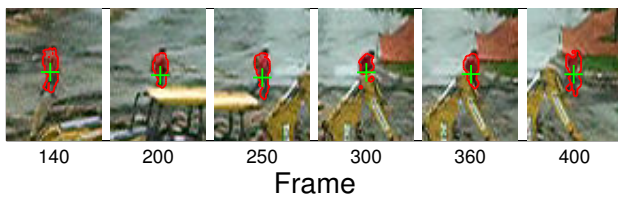
Localization and segmentation: For performing segmentation and localization, three template images were used. Segmentation was achieved by threshold as per Section 3.4. The results are shown in Figure 9. For all the sequences in the figure, the resolution is 320×240 . In Figure 9(a), artificial occlusions are created to gauge the performance of tracker under occlusion. First the upper part of the body is occluded, then the lower part. The tracker successfully tracks the target. For the additional sequences, the localization and segmentation results are equally good.



(a) Sequence 1: resolution 320×240 , window size 25×35 .



(b) Sequence 2: resolution 320×240 , window size 10×20 .



(c) Sequence 3: resolution 320×240 , window size 25×35 .



(d) Sequence 4: resolution 320×240 , window size 25×35 .

Figure 9. Segmentation results.

5. Conclusion

We presented a novel eigenspace-based object tracking algorithm. KPCA was used for the eigenspace representation due to its nonlinear characteristic. A similarity function was derived to measure the similarity of pixel vectors to the learned space. Target localization was carried out using a variational approach. We provide both gradient- and mean shift-based optimization procedures that optimize the similarity function in the eigenspace.

Acknowledgement: The research was supported in part by NSF ECCS #0622006.

References

- [1] O. Arif and P. Vela. Kernel map compression using generalized radial basis functions. In *IEEE ICCV*, 2009.
- [2] S. Avidan. Ensemble tracking. In *IEEE CVPR*, pages 494–501, 2005.

- [3] M. Black and A. Jepson. Eigentracking: Robust matching and tracking of articulated objects using a View-Based representation. *IJCV*, 26(1):63–84, 1998.
- [4] Y. Cheng. Mean shift, mode seeking, and clustering. *IEEE TPAMI*, 17:790–799, 1995.
- [5] T.-J. Chin and D. Suter. Incremental kernel principal component analysis. *IEEE TIP*, 16:1662–1674, 2007.
- [6] D. Comaniciu, P. Meer, and V. Ramesh. Kernel-based object tracking. *IEEE TPAMI*, 25:564–577, 2003.
- [7] C. Ding and X. He. K-means clustering via principal component analysis. In *Proc. of Int. Conf. on Machine Learning*, pages 225–232, 2004.
- [8] A. Elgammal, R. Duraiswami, and L. Davis. Probabilistic tracking in joint feature-spatial spaces. In *IEEE CVPR*, pages 781–788, 2003.
- [9] P. Gabriel, J.-B. Hayet, J. Piater, and J. Verly. Object tracking using color interest points. In *IEEE Conference on Advanced Video and Signal Based Surveillance*, pages 159–164, 2005.
- [10] M. Grabner, H. Grabner, and H. Bischof. Learning features for tracking. In *IEEE CVPR*, pages 1–8, 2007.
- [11] J. Lim, D. Ross, R. Lin, and M. Yang. Incremental learning for visual tracking. In *In Advances in Neural Information Processing Systems*, pages 793–800. MIT Press, 2004.
- [12] L. Lu and G. Hager. A nonparametric treatment for location/segmentation based visual tracking. In *IEEE CVPR*, pages 1–8, 2007.
- [13] J. Meltzer, S. Soatto, M.-H. Yang, and R. Gupta. Multiple view feature descriptors from image sequences via kernel principal component analysis. In *ECCV*, pages 215–227, 2004.
- [14] M. Ozuysal, V. Lepetit, F. Fleuret, and P. Fua. Feature harvesting for tracking-by-detection. In *ECCV*, pages 592–605. Springer, 2006.
- [15] F. Porikli, O. Tuzel, and P. Meer. Covariance tracking using model update based on Lie algebra. In *IEEE CVPR*, pages 728–735, 2006.
- [16] B. Scholkopf, A. Smola, and K.-R. Muller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Comp.*, 10(5):1299–1319, 1998.
- [17] M. Singh, H. Arora, and N. Ahuja. Robust registration and tracking using kernel density correlation. In *IEEE CVPR*, pages 174–174, 2004.
- [18] A. Smola, A. Gretton, L. Song, and B. Scholkopf. A Hilbert space embedding for distributions. *Lecture Notes in Computer Science*, 4755:40–41, 2007.
- [19] S. Soatto and A. J. Yezzi. Deformation: Deforming motion, shape average and the joint registration and approximation of structures in images. *IJCV*, 53:153–167, 2003.
- [20] C. Twining and C. Taylor. Kernel principal component analysis and the construction of non-linear active shape models. In *BMVC*, volume 1, pages 23–32, 2001.
- [21] P. Viola and W. Wells. Alignment by maximization of mutual information. In *IJCV*, pages 16–23, 1995.
- [22] C. Yang and L. D. R. Duraiswami. Efficient mean-shift tracking via a new similarity measure. In *IEEE CVPR*, pages 176–183, 2005.