

Learning to Track: Conceptual Manifold Map for Closed-form Tracking

Ahmed Elgammal

Department of Computer Science, Rutgers University, Piscataway, NJ, 08854 USA
elgammal@cs.rutgers.edu

Abstract

Our objective is to model the visual manifold of object appearance corresponding to geometric transformation. We learn a generative model for object appearance where the appearance of the object at each new frame is a function that maps from a conceptual representation of the geometric transformation space into the visual manifold. By learning such generative model we can infer the geometric transformation (track) directly from the tracked object appearance. As a result tracking can be achieved in a closed form and therefore can be done very efficiently.

1 Introduction

Tracking objects in the visual scene is an essential computer vision task with many applications in robotics, human computer interaction, surveillance, and in medical applications. In the last two decades, extensive research in the computer vision field have focused on problems related to visual motion and tracking. Such interest in tracking is part of wider research that address problems related to time-variant images including detection of motion, quantifying image motion, recovering 3D motion, recognizing motion, modeling object motion, recovering object structure from motion, etc.

Tracking is typically posed as a search problem in a geometric transformation parameter space as well as in body configuration space (for articulated objects) and/or in deformation space (for nonrigid objects). Generally, tracking is based on learning an invariant object representation, then searching for the best fit. Various appearance representations have been used as an invariance for tracking such as texture appearance templates, e.g. [2], or through linear subspace representation [4, 12] or through filter response [14]. Appearance can also be represented through global statistical representations [3, 6]. Given such invariance, tracking is a search problem in the parameter space(s) which can be achieved through a nonlinear optimization framework, e.g. [4], through sampling the space, e.g., [13], maximizing a classifier score as in SVM tracking [1, 17], or in a closed form, e.g. [16] for tracking image batches.

Our goal is to achieve trackers that can directly infer such parameters from object appearance through learned models of the visual manifolds of such parameter spaces. As an object moves in the 3D world different views of the object might be visible and also, the object will appear under different illumination conditions. So, obviously, object appearance through time is a function of view, illumination, articulation, deformation, etc., and, therefore the resulting visual manifold are quite complicated given all these conceptually orthogonal factors. However, attempt has been made for modelling object appearance as function of some of these factors. For example, in [4] different views of the object can be represented using a linear subspace representation using PCA. Similarly, Linear subspace analysis was also shown, empirically and theoretically, to be able to model infinite illumination space as in [10] and others.

Our objective in this paper is to learn a representation of the appearance of the object given a class of geometric transformation in a way that facilitates the recovery of such transformation parameters at tracking time in a closed form. In order to achieve this we need to model the visual manifold of object appearance corresponding to geometric transformation. However this is quite a challenging task given the above mentioned factors. Therefore, in this paper we focus on geometric transformation assuming all other factors are invariant. We consider the tracking problem as a learning problem where a generative model for object appearance is learned off-line. That is, the appearance of the object at each new frame is a function that maps the appearance of the object given a particular geometric transformation to the visual input. This is achieved through learning a smooth nonlinear mapping function from a conceptual representation of the transformation space to the actual visual manifold in a way which facilitates the recovery of the transformation parameter at each new frame. This yields a very efficient closed-form tracking procedure that we applied successfully to track image regions.

The paper is organized as follows: section 2 and 3 describe the learning approach. Section 4 describes how to use the learned model in tracking image regions. Section 5 shows some experimental results and evaluations.

2 Problem Definition

Suppose that we want to track a rigid object that undergoes a geometric transformation which can be parameterized in the image space (e.g., translation, rotation, affine, etc.) and suppose that we are looking through an image window \mathcal{W} that lies inside the object. This window can be large enough to contain the whole object or small around a point feature. Of course, if there is enough texture in \mathcal{W} we can recover the geometric transformation. This is typically posed as a search problem in the geometric transformation parameter space for the optimal parameters that minimize some correlation-based metric. However, in this paper, we try to pose this problem as a learning problem where we learn to recover the transformation parameters directly from the appearance of the image window \mathcal{W} .

The appearance of an image window, such as \mathcal{W} , represents a point in a high dimensional visual input space. If the object undergoes a geometric transformation, the observed intensity at \mathcal{W} will change and this change corresponds to moving along a certain manifold in the visual input space defined by \mathcal{W} . Since the geometric transformation space, e.g. translation, is low in dimensionality, we know that visual manifold corresponding to such transformation is also low in dimensionality. However such manifold is embedded in a high dimensional image space observed through the image window. Based on the geometry and the appearance (texture) of the tracked object, such manifold is nonlinear and might twist and even self intersect. The manifold also can be degenerate to a lower dimension (for example consider the aperture problem). In such case the recovery of the geometric transformation will not be possible.

The bottom line is that the dimensionality of the visual manifold is less than or equal to the dimensionality of the corresponding geometric transformation space (assuming no other factors affecting the appearance.) If we consider the case where the visual manifold is not degenerate, then we can establish a one to one mapping between a conceptual representation of the geometric transformation manifold and the corresponding visual manifold. If such mapping is established, then we can directly infer the geometric transformation from the object appearance.

If we consider learning mappings between a geometric transformation space and a visual input space we have to face the nonlinearity mentioned above. The first question then is what form of mapping can we use? The second question is which direction can we learn the mapping? Since we need to recover the transformation parameters, it might be obvious to map from the visual space to the transformation space. But such mapping is ill-defined, considering the degenerate cases mentioned above. Besides this degeneracy, there are practical issues related to the high dimensionality of such mapping that will be discussed. On the other hand, the geometric transformation space is well-defined

conceptually and, fortunately, if we consider tracking applications, only a very small range of the parameter space is relevant between consecutive frames. So, our goal is to learn a generative model of object appearance as a function of the geometric transformation parameters. This can be achieved through learning a nonlinear mapping from a conceptual representation of the transformation space into the visual input manifold. We learn a nonlinear mapping using a semi-parametric radial basis function formulation in a way similar to [9, 8]. Obviously, such mapping is not necessarily invertible to recover the transformation parameters. So we need to find a good and efficient approximation for the inverse, as well as, to define the criteria where such inverse is good enough.

The third question we address is how can this formulation be useful for tracking? Obviously, if we achieve such learning, the transformation parameters can be inferred directly and the tracking can be done very efficiently. However, can the learning be affordable to be done as an initialization step. This requires that we decompose the learning procedure to parts which do not depend on the tracked object which can be done off-line and parts that can be done efficiently at initialization.

3 Conceptual Manifold Map

Let the visual input space be defined as the intensity observed through an image window \mathcal{W} located inside the object to be tracked and be represented as a vector $y \in \mathbb{R}^d$. If the underlying region (object) undergoes a geometric transformation $T(\cdot; a)$ with parameters a between two frames while \mathcal{W} remain fixed, then the observed intensity through \mathcal{W} is a function of the parameter a . In tracking the object motion is limited between frames to certain domain of the parameter space. Let us denote such domain by Ω and denote the initial parameter as a_o .

Given a set of distinctive representative points $X = \{x_i \in \mathbb{R}^e, i = 1 \dots N\} \subset \Omega$ on the parameter space, where e is the dimensionality of the parameter space, and their corresponding visual inputs $Y = \{y_i = T(y; x_i), i = 1 \dots N\}$ (transformed images observed through \mathcal{W}), we can learn a nonlinear mapping between the conceptual parameter space, represented by X , and visual input manifold, represented by Y . This can be achieved by a set of nonlinear mapping functions $f^k(x)$, one for each pixel k in the visual input, that approximates $f^k(x_i) \cong y_i^k = T(y^k; x_i), i = 1 \dots N$ and minimizes a regularized risk criteria

$$H(f^k) = \sum_{i=1}^N \| f^k(x_i) - y_i^k \| + \lambda \Phi[f^k] \quad (1)$$

where $\Phi[f]$ is a smoothness functional and λ is a regularization parameter. For the purpose of modeling the visual manifold of geometric transformation, we need a measure of smoothness that limits the oscillation of the function which

can be achieved by defining a smoothness functional that limits the energy at high frequency domain, as suggested in [11], in the form

$$\Phi[f] = \int_{R^e} ds |\tilde{f}(s)|^2 / \tilde{\Phi}(s) \quad (2)$$

where $\tilde{\Phi}$ falls to zero as $\|s\| \rightarrow \infty$ and $\tilde{\cdot}$ indicates the fourier transform. A special class of such stabilizers are radial symmetric stabilizers which are invariant under rotation and translation. In particular Duchon [7] multivariate splines uses a smoothness functional of the the form of equation 2 with $\tilde{\Phi} = 1/\|s\|^{2m}$ which yields basis function of the form

$$\phi(x) = \begin{cases} \|x\|^{2m-e} \ln\|x\| & \text{if } 2m > e \text{ and } e \text{ is even} \\ \|x\|^{2m-e} & \text{otherwise} \end{cases} \quad (3)$$

Examples of this family include thin-plate spline ($\phi(u) = u^2 \log(u)$), biharmonic ($\phi(u) = u$) and triharmonic ($\phi(u) = u^3$) splines. Another example of radial stabilizers is stabilizer where $\tilde{\Phi} = e^{-\|s\|/\beta}$ in which case Gaussian function is the basis function and β is a positive scale parameter.

In general if Φ is symmetric it can be shown [7, 5] that functions of the form

$$f^k(x) = p^k(x) + \sum_i^N w_i^k \phi(\|x - x_i\|), \quad (4)$$

minimizes equation 1 where $\phi(\cdot)$ is a real valued basic function, w_i are real coefficients, $\|\cdot\|$ is the norm on R^e , p is a linear polynomial of degree $m \leq e$. We use first degree polynomials, i.e., $p^k(x) = c^{k^T} \cdot [1 \ x]^T$ with coefficients c^k . To insure orthogonality the following constraints (side conditions) are imposed:

$$\sum_{i=1}^N w_i p_j(x_i) = 0, j = 1, \dots, m$$

where p_j are the linear basis of p . Such added constraints make the problem well posed, i.e., we have $N + e$ equations in $N + e$ unknowns.

Given the representative sets of points in the transformation space $\{x_i\}$ and their corresponding visual inputs (transformed images) $\{y_i\}$ the mapping coefficients for pixel k , w_i^k and c^k , can be obtained by solving a system of linear equations

$$\begin{pmatrix} A + \lambda I & P \\ P^T & 0 \end{pmatrix} \begin{pmatrix} w^k \\ c^k \end{pmatrix} = \begin{pmatrix} y^k \\ 0 \end{pmatrix}, \quad (5)$$

where $A_{i,j} = \phi(\|x_j - x_i\|)$, $i, j = 1 \dots N$, P is a matrix with i -th row $[1 \ x_i^T]$, $w^k = (w_1^k \dots w_N^k)^T$ and $y^k = [y_1^k \dots y_N^k]^T$ is $N \times 1$ vector containing pixel k for

each of the transformed images. The whole mapping can be written in a matrix form as

$$f(x) = B \cdot \psi(x) \quad (6)$$

where B is a $d \times (N+e+1)$ dimensional matrix with the k -th row $[w_1^k \dots w_N^k \ c^{k^T}]$ and $\psi(x)$ is the $N+e+1$ dimensional vector

$$\psi(x) = [\phi(\|x - x_1\|) \dots \phi(\|x - x_N\|) \ 1 \ x^T]^T. \quad (7)$$

Therefore the solution for B can be obtained by directly solving the linear systems

$$\begin{pmatrix} A + \lambda I & P \\ P^T & 0 \end{pmatrix} B^T = \begin{pmatrix} Y \\ 0_{(e+1) \times d} \end{pmatrix}, \quad (8)$$

where Y is $(N \times d)$ matrix containing the transformed images, i.e., $Y = [y_1 \dots y_N]^T$. In terms of kernel learning theory, equation 6 shows that a linear mapping can be established between a kernel induced space defined by the empirical kernel map [15] in equation 7 given the set of points X .

Notice, that for learning such mapping from the conceptual transformation space into the visual input space, the left hand side matrix in equation 8 does not depend on the tracked region since both A and P depends only on the the positions of the points $\{x_i\}$ in the transformation space. Such matrix and its inverse can be computed off-line given a fixed points in the transformation space and therefore solution of the multiple systems will only reduce to a matrix multiplication given the right hand side.

4 Tracking

4.1 Solving for the Inverse

The generative model defined by the nonlinear mapping in equation 6 maps from a conceptual transformation space to a visual input space, i.e., it can be used to synthesize (interpolate) the observed intensity at image window \mathcal{W} given transformation parameter x within the domain Ω used in the learning. However, we are interested in tracking, i.e., recovering transformation parameters x given the observed intensity in image window \mathcal{W} . Therefore, we are interested in the inverse of the mapping, i.e., a discriminative model (mapping from the visual input to the transformation space). One option is to learn such inverse mapping using the same framework described in section 3. However, this is not feasible for the following reasons:

1. The inverse mapping is a mapping from \mathbb{R}^d to \mathbb{R}^e where $d \gg e$, i.e., a mapping from a high dimensional space. Without enough sampling of the input space, such inverse mapping will not be constrained and therefore will not be able to interpolate given new images. However, learning the generative mapping

(from the transformation space to the input space) is feasible and can generalize since it involved multiple mappings from a low dimensional transformation space to each individual pixel i.e., from \mathbb{R}^e to \mathbb{R} .

2. Learning the inverse mapping in the form of equation 8 will require computation of the matrix A for each tracked target region which is not practical. Learning generative mapping did not require such computation since the matrix A does not depend on the target region in this case.
3. The inverse mapping is not necessarily a function if we consider degenerate cases (the aperture problem).

For these reasons, our framework is based on computing an approximate inverse for the direct mapping instead of learning the inverse mapping itself. Given the observed intensity $y \in \mathbb{R}^d$ through the image window, it is required to find the corresponding transformation $x \in \mathbb{R}^e$ by solving for the inverse of the mapping. Each input yields a set of d nonlinear equations in e unknowns (or one e -dimensional unknown). Therefore a solution for x can be obtained by solving the nonlinear system

$$x^* = \arg_x \min \|y - B\psi(x)\| \quad (9)$$

However because of the linear polynomial part in the interpolation function in equation 4, the vector $\psi(x)$ has a special form that facilitates an efficient least square linear closed form approximation for x .

Notice that the rank of the matrix B is at most N since there are N RBF centers. The rank of B can be less than N if the mapping is degenerate which can happen if the window \mathcal{W} has no enough texture to recover the transformation x . In this case different points in the transformation space maps to the same visual input (aperture problem). This issue will be discussed in the next section. Therefore, Moore-Penrose generalized-inverse of the matrix B can be obtained by decomposing the matrix B using SVD such that $B = USV^T$ where U is $d \times (N + e + 1)$ and V is $(N + e + 1) \times (N + e + 1)$ and S is a diagonal $(N + e + 1) \times (N + e + 1)$. The vector $\psi(x)$ can be recovered simply by

$$\psi(x) = V \hat{S} U^T y$$

where \hat{S} is the diagonal matrix obtained by taking the inverse of the nonzero singular values in S the diagonal matrix. However, because of the special form of the vector $\psi(x)$ we only need the last e elements of the vector. Therefore, a least square linear approximation for x can be recovered by considering linear weight matrix C such that

$$x = C \cdot y \quad (10)$$

where C is an $e \times d$ matrix obtained by taking the last e rows from the pseudo-inverse $(U \hat{S} V^T)^T$. Equation 10 shows a

closed form solution to obtain the geometric transformation x as a linear weighted sum of the observed intensity in image window \mathcal{W} . Notice that such linear weights are approximate solution to equation 9. This is different from directly fitting a linear model between the transformation space and the visual input space. This will be shown in section 5.

4.2 Tracking Procedure

The basic assumption in tracking is that the geometric transformation for tracked target region will be small between consecutive frames. That will limit the search space for the geometric transformation parameters (displacement) at each new frame. In our framework, there is no explicit search required at each new frame. Instead, given a learned nonlinear mapping from a geometric transformation space into a visual input space represented by an image window, at each new frame, the appearance of that image window is used to drive the geometric transformation parameters in a closed form using the learned tracking weights in equation 10. If the ‘‘displacement’’ between frames is within the learned parameter domain then such displacement can be recovered. Notice that the inverse mapping solution obtained above facilitates for sub-pixel interpolation for the displacement or for the geometric transformation parameters in general. Given the recovered parameters, the target window is then warped using these parameters to compensate for such transformation which will retain the appearance of the window as the mapping of the coordinate center of the geometric transformation domain, a_o . We can summarize the tracking procedure as follows:

Off-line: Given a set of representative points $x_i, i = 1 \dots N$ in the geometric transformation space, compute A, P

Pre-tracking (learning): Given image window \mathcal{W}

- Compute $y_i = T(y; x_i)$
- Compute the coefficient matrix B
- Compute SVD of B
- Compute the weights C

Tracking: Given initial window \mathcal{W}_{t_0}

For each new frame t

- $y_t =$ observed intensity through \mathcal{W}_{t-1}
- Compute transformation $x_t = C \cdot y_t$
- Warp image window coordinates:
 $\mathcal{W}_t = T(\mathcal{W}_{t-1}; x_t)$

The tracking can be done very efficiently since it involves only one matrix multiplication per frame. The main computational cost of the tracking is in the pre-tracking step which is performed only at initialization. The pre-tracking

involves three steps: 1) compute transformed image window which can be performed very efficiently. For example, if we consider translation only, then any transformed image can be obtained just by indexing the original image at a displaced window, i.e., no actual computation is required. Learning other geometric transformation such as rotations or affine will require warping the image window. 2) The computation of the coefficient matrix B reduces to only a matrix multiplication since the left hand side matrix and its inverse is computed off-line. 3) The main computational cost is the SVD decomposition of B . Note that only the first $(N + e + 1)$ columns of the U matrix are needed. It is important to mention that the tracked window can be of any shape and that using color or other image features to describe the appearance is straight forward.

4.3 Tracking Example: Tracking a Rigid Object

To illustrate a learning example, we use the coke pick up sequence from [4] as an example of a rigid body tracking. Figure 1-a shows the initial frame and the image window used in the learning. We learn a nonlinear mapping from the space of translations and rotation to the visual input represented by the marked image window. The centers for the RBFs were chosen at $-4, -2, 0, 2, 4$ pixels for both the horizontal and vertical translations and at $-2, -1, 0, 1, 2$ degrees for the rotation, i.e., 125 RBF centers were used. Figures 1-b,c,d show the learned tracking weights for the horizontal, vertical displacements and the rotation respectively where bright and dark pixels indicates high magnitude weights with positive and negative signs respectively. As apparent from the figures, the learned weights are correlated with image features. For example, the weights that drive the horizontal displacement are oriented with vertical edges. Similarly the weights that drive the vertical displacement are oriented with horizontal edges. The weights that drives the rotation are correlated with edges away from the center of the window. Figure 2 shows the tracking result. The tracker can successfully recover the translation and rotation parameters in closed-form for the entire sequence. Notice that the tracker is robust to partial occlusion. The coke can is partially visible in the image in the last 30 frames of the sequence, however, the tracker can correctly estimate geometric transformation from the visible part. Further analysis of this example will be shown in section 5.

5 Experimental Result

In this section we show some tracking results using the proposed framework for tracking image regions as well as some comparisons with other learning approaches. We already have shown an example for rigid object tracking using the coke sequence in section 4.3 and figure 2. Here we elaborate on this example to show the robustness of the tracker in presence of image noise and to compare it to alternative mapping approaches. We compare four possible

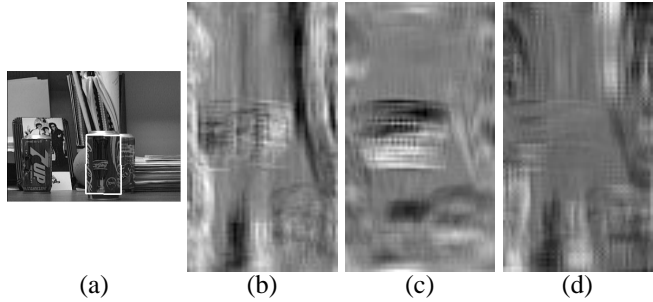


Figure 1. a- Image window used in learning. b,c,d- obtained weights for horizontal, vertical displacement and rotation respectively

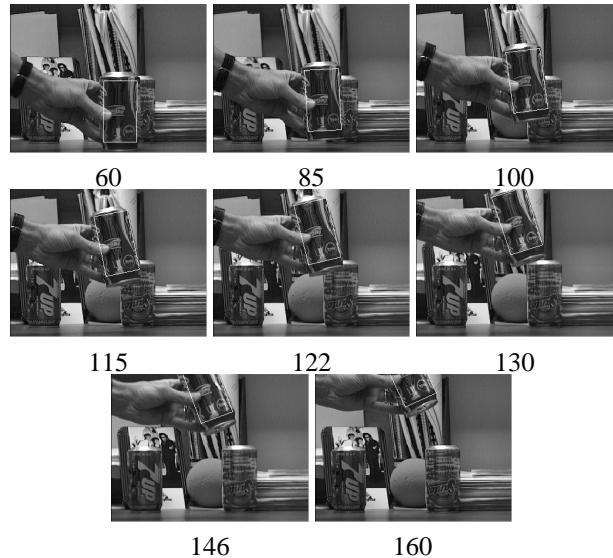


Figure 2. Tracking the coke can by recovering translation and rotation in a closed form using the learned linear weights shown in figure 1. Notice, the tracking is robust to partial occlusion as in frame 146,160

forms of mappings between the conceptual manifold representation and the visual input space. In all the cases we have an initial image window (figure 1-a) represented as $y \in \mathbb{R}^d$ that we used to obtain a set of transformed images $\{y_i = T(y; x_i), x_i \in \mathbb{R}^e, i = 1 \dots N\}$ using transformation parameters x_i . We also generated different evaluation sets of noisy transformed images using new disjoint set of transformation parameters $\{x_j, j = 1 \dots M\}$ and adding Gaussian noise $\delta_\sigma \approx \mathcal{N}(0, \sigma^2)$ at each pixel with varying σ in the range 0 to 50. i.e., the evaluation sets are $\{y_j^\sigma = T(y; x_j) + \delta_\sigma, j = 1 \dots M, \sigma = 0 \dots 50\}$. We used 49 points for learning and 120 points for evaluation all representing translation in the range $-6,6$ pixels. Given the set $\{(x_i, y_i)\}$ we fit four different forms of mapping as follows:

1- Generative Nonlinear Map: This is the approach proposed in this paper where the mapping is from the conceptual manifold representation to the visual input in the form

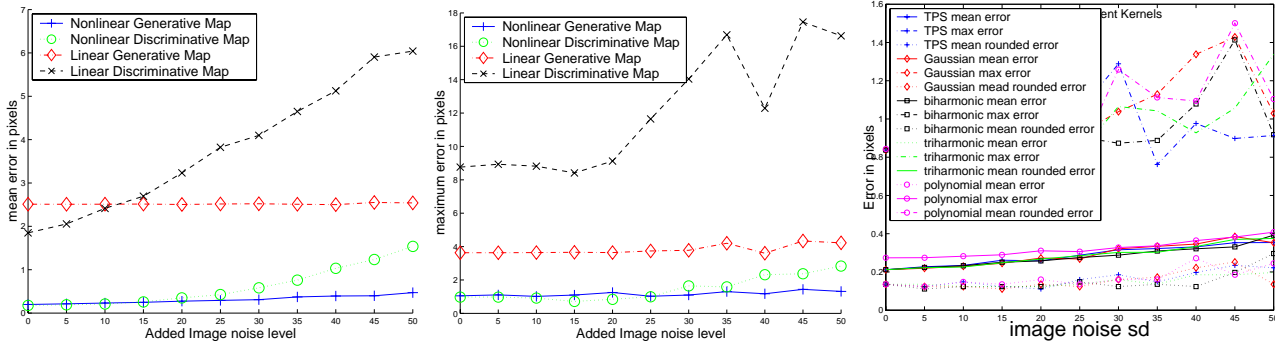


Figure 3. Error in recovering geometric transformation with different mappings as a function of image noise.

of equation 6 and the tracking is performed in closed form using the weights obtained in equation 10.

2- Discriminative Nonlinear Map: This is a nonlinear map from the visual input to the conceptual manifold representation in the form of $x_i = B\psi(y_i)$ fitted in the same way described in section 3. In this case the tracking is done directly by evaluating the mapping at each evaluation image y_j .

3- Generative Linear Map: This is a linear map from the conceptual manifold representation to the visual input in the form $y_i = Bx_i$ where B is a $d \times e$ matrix. The tracking is performed by computing $x = B^+y$ using the pseudo inverse B^+ .

4- Discriminative Linear Map: This is a linear mapping from the visual input into the conceptual representation in the form $x_i = By_i$.

All the four mapping approaches are evaluated using the noisy image sets $\{\hat{x}_j, \hat{y}_j^\sigma\}$. The error metric used is the mean distance between estimated transformation parameters and ground truth parameters. Figures 3-a,b show the mean error and the maximum error for each of the learning approaches over the different evaluation sets. As can be noticed from the figures, the proposed nonlinear generative map with approximate closed-form inverse solution has the least error and is the most robust to noise even with high level of image noise, e.g., $\sigma = 50$. Notice also, that mean error is less than one pixel, i.e., we can reach sub-pixel estimate of the transformation parameters. We also evaluated different kernels for the conceptual manifold map including polynomial kernels, Thin plate spline, Gaussian, biharmonic, and triharmonic kernels. Figure 3-c shows the obtained errors (mean error, maximum error, mean rounded error) using each of the kernels. As can be noticed from the figure, all kernels show qualitatively similar results.

Figure 4 shows results for tracking a face. The person head is moving as well as the camera. The tracker can successfully recover translation and rotation in closed form for about 800 frames. The tracker finally failed when the camera moved abruptly such that the transformation between frames fell out of the learned range.

Figure 5 shows tracking two blobs (head and torso) of a walking person. The tracker successfully recover the transformation parameters in a closed-form for the whole se-



Figure 4. Tacking a Face: recovering translation and rotation in a closed-form for over 800 frames.

quence. Notice that, for the torso blob, the hand articulation changes the appearance of the tracked window, yet the tracker is robust to such changes.

Figure 6 shows an example of tracking a full body. There is a lot of variation in person appearance through the sequence due to articulation and changing pose w.r.t. the camera. The tracker can successfully recovers translation parameters in a closed-form despite the big changes in appearance relative to the small size of the tracked window.

6 Conclusion & Future Work

In this paper we showed how can we recover the geometric transformation parameters between frames in closed-form by learning a nonlinear generative model of object appearance which maps from a conceptual representation of the geometric transformation manifold and the actual visual

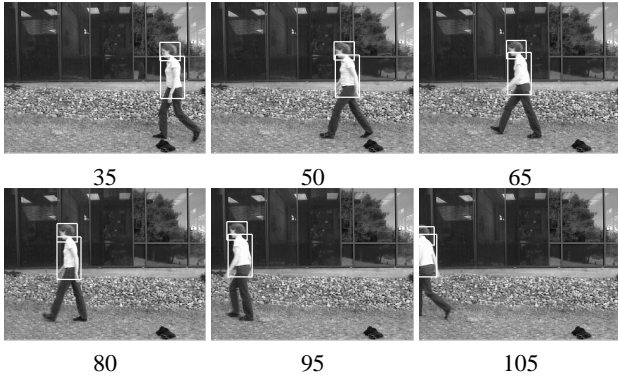


Figure 5. Tacking two blobs in 'walking straight' sequence

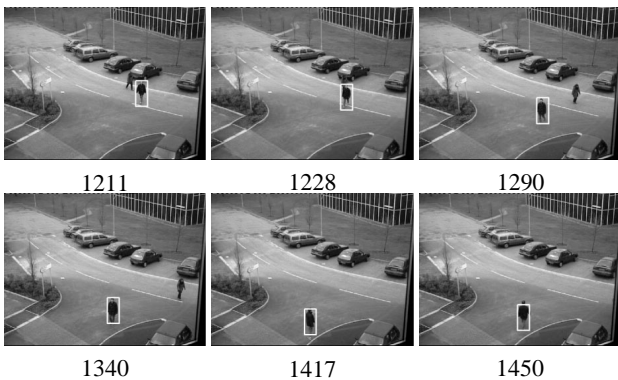


Figure 6. Tacking a human in pets2000 sequence.

input space. The resulting tracker was shown to be robust to image noise, partial occlusion, and changes in appearance due to articulation. The tracker was compared to alternative generative and discriminative learning approaches to achieve the same task of mapping between the conceptual representation and the input space. The proposed method yields the least error and is the most robust to image noise w.r.t. the other approaches in the comparison.

One of the limitation of the proposed approach is the inability to recover between-frame transformations if they fall out of the learned range. However, this gap can be closed by performing tracking at multi-resolutions which can compensate for wider range of transformation at the higher levels. Another alternative, is to combine other search-based tracker to recover from failure. Certainly, any closed-form solution will be suspect to image noise and other uncertainty factors. Nevertheless, any closed-form solution provides a good initial point for any robust estimation framework. Our future objective is to combine the proposed approach to model the geometric transformation manifold with other factors affecting the appearance such as different views and different illuminations. As pointed out in the introduction linear subspace analysis was shown to be able to model multiple views and to model the illumination subspaces. This suggests that views and/or illumination can be combined

through subspace analysis of the nonlinear mapping coefficient in a way similar to [9]. Alternative approach is to combine multiple spaces using tensor-product interpolation [5].

Acknowledgment This research is partially funded by NSF award IIS-0328991

References

- [1] S. Avidon. Support vector tracking. In *CVPR01*, pages I:184–191, 2001.
- [2] B. Basclé and R. Deriche. Region tracking through image sequences. In *Proc. ICCV*, pages 302–307, 1995.
- [3] S. Birchfield. Elliptical head tracking using intensity gradients and color histograms. In *Proc. CVPR*, Jun 1998.
- [4] M. J. Black and A. D. Jepson. Eigentracking: Robust matching and tracking of articulated objects using a view-based representation. In *ECCV (1)*, pages 329–342, 1996.
- [5] W. Cheney and W. Light. *A course in Approximation Theory*. Brooks/Cole publishing company, 2000.
- [6] D. Comaniciu, V. Ramesh, and P. Meer. Real-time tracking of non-rigid objects using mean shift. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 142–149, Jun 2000.
- [7] J. Duchon. Splines minimizing rotation-invariant seminorms in sobolev spaces. In W. Schempp and K. Zeller, editors, *Constructive Theory of Functions of Several Variables*, number 571 in Lecture Notes in Mathematics, pages 85–100, 1977.
- [8] A. Elgammal. Nonlinear generative models for dynamic shape and dynamic appearance. In *Proc. of 2nd International Workshop on Generative-Model based vision. GMBV 2004*, July 2004.
- [9] A. Elgammal and C.-S. Lee. Separating style and content on a nonlinear manifold. In *Proc. of CVPR*, June-July 2004.
- [10] A. Georghiades, D. Kriegman, and P. Belhumeur. Illumination cones for recognition under variable lighting: Faces. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, 1998.
- [11] F. Girosi, M. Jones, and T. Poggio. Priors stabilizers and basis functions: From regularization to radial, tensor and additive splines - mit ai lab tech report. Technical Report AIM-1430, 1993.
- [12] G. D. Hager and P. N. Belhumeur. Efficient region tracking with parametric models of geometry and illumination. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 20(10):1025–1039, 1998.
- [13] M. Isard and A. Blake. Contour tracking by stochastic propagation of conditional density. In *ECCV (1)*, pages 343–356, 1996.
- [14] A. Jepson, D. J. Fleet, and T. F. El-Maraghi. Robust on-line appearance models for visual tracking. In *CVPR*, volume I, pages 415–422, 2001.
- [15] B. Schölkopf and A. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization and Beyond*. The MIT Press, Cambridge, Massachusetts, 2002.
- [16] C. Tomasi and T. Kanade. Shape and motion from image streams under orthography: A factorization method. *IJCV*, 9(2):137–154, November 1992.
- [17] O. Williams, A. Blake, and R. Cipolla. A sparse probabilistic learning algorithm for real-time tracking. In *ICCV, Nice, France, Oct. 2003*.