Boston University Computer Science Technical Report No. BUCS-TR-2009-006

Active Hidden Models for Tracking with Kernel Projections

Samuel Epstein and Margrit Betke
Department of Computer Science
Boston University
{samepst, betke } @cs.bu.edu, www.cs.bu.edu/groups/ivc

Abstract

We introduce Active Hidden Models (AHM) that utilize kernel methods traditionally associated with classification. We use AHMs to track deformable objects in video sequences by leveraging kernel projections. We introduce the "subset projection" method which improves the efficiency of our tracking approach by a factor of ten. We successfully tested our method on facial tracking with extreme head movements (including full 180-degree head rotation), facial expressions, and deformable objects. Given a kernel and a set of training observations, we derive unbiased estimates of the accuracy of the AHM tracker. Kernels are generally used in classification methods to make training data linearly separable. We prove that the optimal (minimum variance) tracking kernels are those that make the training observations linearly dependent.

1. Introduction

The design of visual tracking methods is a fundamental goal of computer vision. Tracking methods are needed to solve important video understanding problems such as interpreting human activities for video-conferencing, surveillance, human-computer interaction, sign-language recognition, and driver assistance. In this paper, we introduce a new approach to visual tracking that differs significantly from prior work described in the computer vision literature. Our approach is based on techniques from classification theory, in particular, kernel projections. We show how this fundamental technique can become extremely valuable in the new context of tracking.

Our exemplar based method tracks rigid or deformable objects by uncovering their hidden states. We introduce Active Hidden Models (AHMs) to describe the feasible object states in the hidden state space. We train a Active Hidden Model in advance of the tracking process by applying a kernel function to a large set of training images. Using a kernel function, we can map the training images, which are considered points in an observation space, to their corresponding points in the feature/state space. We select training images that show the objects of interest in representative rigid or nonrigid transformations. We then define tracking as the problem of identifying the image in the observation space that best describes the rigid or non-rigid object transformation seen in the current video frame. To find the best matching image, we map the current object image into the hidden space via the same kernel function and project the resulting point onto the subspace that defines the Active Hidden Model of the object. We then reconstruct a fundamental appearance of this hidden point, resulting in the best-effort reconstruction of the object.

To design a practical tracking system that runs in real time, we introduce a method to approximate the kernel projection. Our experiments show that only a small subset of the training set ($\sim 1/10$) is needed to approximate a projection accurately. In some of our experiments, a subset of size of 2 is sufficient for an accurate projection.

We provide unbiased estimates of the accuracy of our AHM tracking approach given a kernel function and a training set. We show that the optimal (minimum variance) kernels are those that make the training data linearly dependent in the state space.

We successfully applied our AHM-based tracking method to three problem domains – recovering face pose and location, understanding facial expressions, and tracking the deformations of candy wrapper. For each domain, we show that our tracking method can recognize the object transformations accurately and in real time.

The computer vision literature includes various tracking approaches that also leverage results of classification theory; for example, tracking can be interpreted as a model-based binary classification problem which decides whether a region in the current image matches the object model or background [1, 16]. In contrast, our approach does not use a decision function.

Approaches that use kernels for tracking have received much attention (e.g., [4, 10]). The way kernels are applied in these previous works is very different from our approach and it is important to note the difference. In “kernel-based tracking” [4], kernels are convolved with image data to provide spatial smoothness. In contrast, we use a kernel function to map image data of an object into a hidden state space and thus instantiate the Active Hidden Model of the object. Comaniciu et al. [4] used kernels to spatially mask histogram-based object representations, such as color. Dewan and Hager [10] proposed the use of a family of kernel-based sampling functions to produce object-specific kernel configurations. Han et al. [8] used kernels to better approximate the posterior densities propagated in particle filtering.

As in “eigentracking,” e.g., [3], our method uses a least-squares approach to find an approximate representation of the object of interest. Instead of projecting the current measurement onto a select number of principal components, we approximate the projection onto the entire subspace spanned by the training data in the state space.

Our approach is not an extension of deformable template matching (e.g. [17]) because we do not use explicit parameterization of the object transformation. Our approach therefore also differs from Active Blobs [14], Active Shape Models (ASM) [6] and Active Appearance Models (AAM) [5], which use parameterized models of object features, such as polygons and textures. These approaches track instantiations of their models by projecting the current frame onto the space spanned by the principal components of the respective features. In contrast to Active Blobs, ASMs, and AAMs, which are *explicit* models, Active Hidden Models are represented by the linear span of the training images in a *hidden* state space. Like the Active Blob, ASM, and AAM approaches, instantiations of AHMs are computed by a projection from real-time video. Unlike Active Blobs, ASMs and AAMs, these instantiations are not directly accessible.

De La Torre and Frade [9] recently proposed to extend parameterized appearance models (AAMs, etc.) so that the non-linear structure of the data can be represented. Their approach uses Kernel Principal Component Analysis (e.g.,[15]), which must rely on the whole data set. In contrast, our approach only requires the use of a small subset of training images to yield an accurate approximation of kernel projection onto the linear subspace that defines the AHM of the object. We summarize our contributions and their novelty:

- We introduce Active Hidden Models that utilize kernel methods traditionally associated with classification.
- We introduce the approach of subset projection to enable real-time tracking.
- We describe a method for tracking with Active Hidden Models that uses subset projections and factored sampling.
- We provide a theoretical description of the variance of the kernel project and prove that the optimal (minimum variance) tracking kernels are ones that produce linear dependency in the training data. This is in contrast to the usage of kernels in classification theory which traditionally aims to make training data linearly separable.
- Our experiments show that Active Hidden Models are efficient and practical for tracking rigid and deformable objects.

2. Methods

Our approach involves an offline training phase and an online test phase, i.e., the tracking phase. In the training phase, we record the object to be tracked while it is moving through space or deforming in some typical manner. From this video, we choose representative sub-images that show the object in different positions, orientations, deformations, or other conditions. The subimages serve as templates $Q = \{q_i\}_{i=1,\dots,n}$ in the observation space Z, which are transformed into the hidden state space by a mapping ϕ via the use of a kernel (Fig. 1). The span of the states $\{\phi(q_i)\}_{i=1,\dots,n}$ defines the Active Hidden Model of the object.

In the tracking phase, the object sub-image z is first mapped to a point $\phi(z)$ in the hidden state space, and from there projected onto the subspace that is spanned by the Active Hidden Model to yield a hidden state x. The appearance of this hidden state $\phi^{-1}(x)$ can be approximated by $\hat{a}$. The reconstructed image $\hat{a}$ can be used for positional tracking in the original video.

In Sec. 2.4, we describe how to use Active Hidden Models for tracking. Our algorithm goes from computing the observation z to its mapping $\phi(z)$, to its hidden state x, and finally to the approximate appearance $\hat{a}$ of the hidden state x. To explain this algorithm, we first give a background on kernel projections from machine learning [12] (Sec. 2.1). We then describe our own contribution of the subset projection (Sec. 2.2) and show how to construct an approximate appearance $\hat{a}$ of the hidden state x (Sec. 2.3).

2.1. Background on Kernel Projections

Given is a set of observations $Q = \{q_i\}_{i=1,\dots,n}$ from observation space Z. There is a mapping ϕ from observation space Z to state space $F = \mathbb{R}^l$, where l could potentially be infinite. The kernel function $k : Z \times Z \to \mathbb{R}$ is the inner product of this mapping:

$$k(q, q') \quad = \quad \phi(q)^T \phi(q').$$

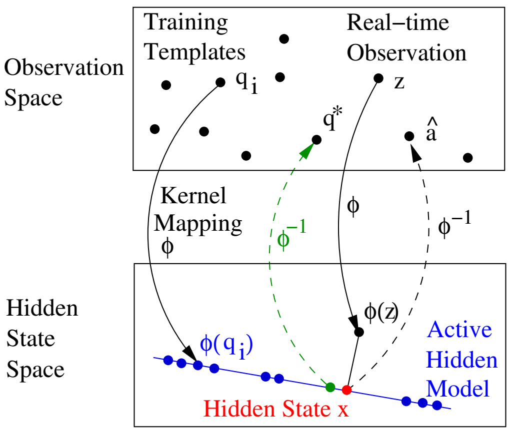

Figure 1. Overview of Approach. We map template images of the object (black points) from the observation to the hidden state space to create the Active Hidden Model (AHM) of the training set (blue line). During tracking, the hidden state x (red) of the current object observation z is computed by projecting $\phi(z)$ onto the AHM. An approximation of the appearance of the hidden state, $\hat{a} \approx \phi^{-1}(x)$, can be created using kernel-dependent methods and the training set. The hidden state (green) closest to x is that of the best-matching template image q^*.

The kernel trick allows ϕ to be implicitly defined by kernel k, so that the mapping ϕ is "hidden." The Gram matrix K is an $n \times n$ matrix, whose (i, j)th element is $k(q_i, q_j)$. The design matrix Φ is an $l \times n$ matrix whose ith column is $\phi(q_i)$. It follows that $\Phi^T \Phi = K$. The empirical map is defined by $k : Z \rightarrow \mathbb{R}^n$, where the ith value of vector $k(z)$ is $k(q_i, z)$.

Given an observation $z \in Z$, the kernel projection maps $\phi(z)$ to its hidden state x of observation z. This hidden state x is the closest point in the linear span of Q in F to $\phi(z)$. Although x is not explicitly defined, it can be represented as the linear combination of the elements of Q:

$$x \quad = \quad \sum_{i=1}^{n} \alpha(i)\phi(q_i) = \Phi\, \alpha,$$

where $\alpha(i)$ is the ith element of the vector α that represents the coordinates of x using Q as a basis. The kernel projection is of the form [12]

$$\alpha = \Phi^{+}\phi(z) = \left(\Phi^T \Phi\right)^{-1} \Phi^T \phi(z) = K^{-1}k(z), \qquad (1)$$

where the term Φ^{+} represents the pseudoinverse of the design matrix. The Gram matrix K might be singular, but the limit $\lim_{\delta \rightarrow 0}(K + \delta I)^{-1}k(z)$ is guaranteed to exist by the definition of pseudoinverses.

The coordinates α can be used to compute the squared distances d_i^2 from the hidden state x to each training object $q_i \in Q$ in F:

$$\begin{aligned} d_i^2 &= \|\phi(q_i) - x\|^2 = \|\phi(q_i) - \Phi\, \alpha\|^2 && (2) \\ &= k(q_i, q_i) - 2k(z, q_i) + \alpha^t K \alpha. && (3) \end{aligned}$$

The term K_i represents the ith element of K. We define the weights w_i of each training set with respect to observation z and kernel k to be the inverse of the squared distance of z to q_i:

$$w_i \quad = \quad d_i^{-2} \qquad (4)$$

The weights are important because they represent the level of similarity between the hidden state x and each training object q_i. This similarity will be leveraged in Sec. 2.3 to recreate the approximate appearance of x.

2.2. Subset Projection

The computation complexity of the kernel projection is on the order of $|Q|$, which may be prohibitively expensive if the training set size is large. We introduce the subset projection, which approximates the kernel projection using only a subset $R \subseteq Q$, where $R = \{r_i\}_{i=1,\ldots,m}$ and $m \ll n$. The first step is to project $\phi(z)$ onto the linear span of R to yield

$$\beta = K_R^{-1} k_R(z), \qquad (5)$$

which represents a vector of size m, the projection coordinates of z with respect to R. The terms K_R and k_R are the Gram matrix and empirical map with respect to R and k. We then solve the reduced subset problem [12]:

$$\min_{\beta} \left\| \sum_{i=1}^{n} \alpha_i \phi(q_i) - \sum_{i=1}^{m} \beta_i \phi(r_i) \right\|^2$$

by taking the derivative with respect to β and setting it to 0. This yields

$$K_{RQ}\alpha = K_R \beta,$$

where K_{RQ} is the $m \times n$ matrix whose (i, j)th element is $k(r_i, q_j)$. Instead of taking the (pseudo)inverse of K_R, which solves the reduced subset problem, we substitute β with Eq. 5 and solve for α:

$$\begin{aligned} K_{RQ}\alpha &= K_R K_R^{-1} k_R(z) && (6) \\ \hat{\alpha} &= K_{RQ}^{+} k_R(z). && (7) \end{aligned}$$

We will show experimentally the subset projection $\hat{\alpha}$ of Eq. 7 gives an accurate approximation of the kernel projection α for small subsets R with $|R| \approx |Q|/10$.

2.3. Active Hidden Models

The Active Hidden Model is the linear span of the training set Q in the state space implicitly defined using a kernel k. Real-time observations z can be projected onto AHMs using the kernel or subset projections as described in Eqs. 1 and 7:

$$\hat{\alpha} = K_{RQ}^{+} k_R(z) \approx K^{-1} k(z).$$

The next step is to determine the appearance of x. One might argue that if x is the hidden state of observation z, then naturally z is the appearance of x. However we wish to reconstruct a *fundamental appearance*, a, which maps directly to this state, defined by

$$\phi(a) = x.$$

Since x is hidden, the appearance is not readily determined. The challenge of determining such an approximation is known as the pre-image problem in the kernel literature [13].

Using Eq. 2, the distance d_i of the hidden state x to each training image q_i can be computed from the projection coordinates $\hat{\alpha}$. One solution to the pre-image problem is to select the training image q^* closest to x as $\hat{\alpha}$. Another option is to approximate the pre-image of x using the weights w of the training set, computed by Eq. 4. This reconstruction depends on the choice of the kernel used. An example of a kernel that we used in our experiments, the threshold kernel, is given in Fig. 2 The kernel first performs thresholding of a pair of greyscale images according to threshold τ to produce two processed binary observations. The final output is the size of the intersection of the "1" pixels of these two processed observations. Thus, the thresholding kernel is an intersection kernel.

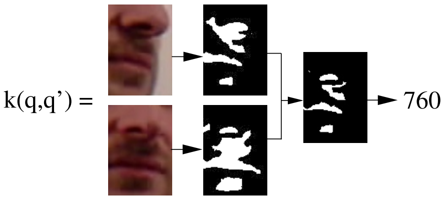

Figure 2. The application of the threshold kernel with examples of greyscale observations q and q', their thresholded versions and intermediate intersection image.

Given the weights w produced using a threshold kernel with threshold τ, the APPEARANCEAPPROXIMATION algorithm (see pseudocode below) computes the approximate appearance $\hat{a}$ of the hidden state x:

$$\phi(\hat{a}) \approx x.$$

The binary image output $\hat{a}$ is created by iterating through every pixel position of the training images and setting a temporary value δ to 0. If the grayscale value of training image q_i is greater than threshold τ at the current position index $(x_{\text{pos}}, y_{\text{pos}})$, then it will "vote" for a 1 pixel by adding weight w_i to δ. Similarly w_i will be subtracted from δ if its intensity is below threshold τ. The contribution of each training sample q_i to the construction of $\hat{a}$ is proportional to w_i. After all training images have voted, the output $\hat{a}$ at position $(x_{\text{pos}}, y_{\text{pos}})$ will have intensity 1 if $\delta \geq 0$, otherwise 0.

function APPEARANCEAPPROXIMATION(Q, τ, w)
    for $x_{\text{pos}} = 1$ to width of training images **do**
        for $y_{\text{pos}} = 1$ to height of training images **do**
            $\delta = 0$
            for $i = 1$ to n **do**
                if $q_i(x_{\text{pos}}, y_{\text{pos}}) \geq \tau$ **then**
                    $\delta = \delta + w_i$
                else
                    $\delta = \delta - w_i$
            if $\delta \geq 0$ **then** $\hat{a}(x_{\text{pos}}, y_{\text{pos}}) = 1$ **else** 0
    return $\hat{a}$

2.4. Tracking with AHMs

We use the kernel projection and appearance recovery method described above to track a deformable object in real-time video. In this section we provide pseudo code of our tracking algorithm, called AHM-TRACKER.

The input to the AHM-TRACKER is the training set Q, a kernel function k, the corresponding Gram matrix K, and its (pseudo)inverse. The training set consists of a group of representative images of the deformable object of interest. The **GetInitPos** and **GetInitWeights** functions leverage apriori knowledge of the application to produce the starting configuration of the tracker. This approach is similar to that of parameterized models (AAMs etc.), whose starting states are usually outside the model formulation. For our implementation of the AHM-TRACKER, the initial position and weights are decided using an exhaustive search over the whole training set to find the training image that best correlates with the start frame.

The **GetVideoFrame** function returns the complete image frame at the current time t. The **GetRealTimeObs** function crops a subimage located at the current position p from the current video frame v. This subimage is observation z.

function AHM-TRACKER(Q, K, k)
    AHM-HELPER(0,GetInitPos(),GetInitWeights())

function AHM-HELPER(t,p,w)
    v = GetVideoFrame(t)
    z = GetRealTimeObs(v, p)
    R = RandomSubset$_{(Q)}(w)$
    $\hat{\alpha}$ = SubsetProject$_{(Q,k)}(z, R)$
    for $i = 1$ to n **do**
        d_i^2 = SquaredDistance$_{Q,k}(z, \hat{\alpha})$
        $w'(i) = d_i^{-2}$
    $\hat{a}$ = APPEARANCEAPPROXIMATION$_{(Q,\tau)}(w')$
    p' = PositionalSearch$(v, p, \hat{a})$
    Output$(t, p', w', \hat{a})$
    AHM-HELPER$(t + 1, p', w')$

The **RandomSubset** method produces a small subset $R \subset Q$ of size $m \ll n$. Each member of R is drawn randomly from Q according to the current weights w. Thus training images that were close to the hidden state of the object in the previous frame have a larger chance of being selected for the current subset R. There are no duplicate elements in R. Once R has been determined, the method **SubsetProject** produces the subset projection of z using R and Q. To compute K_{RQ}^{+} of Eq. 7, rows of K corresponding to the elements of R are copied into a new matrix and its pseudoinverse is computed.

When the coordinates $\hat{\alpha}$ have been created, they are used to compute the distances in the state space and then a set of weights w' using Eq. 3. The approximate fundamental appearance $\hat{a}$ of the hidden state is computed by APPEARANCEAPPROXIMATION, which takes as inputs the weights w' and the training set Q.

The AHM-TRACKER uses the APPEARANCEAPPROXIMATION algorithm (Sec. 2.3), which outputs binary image $\hat{a}$ if the threshold kernel is used. The **PositionSearch** computes the optimal local alignment p' of $\hat{a}$, given the current video frame v and the previous position p.

The **Output** of the AHM-TRACKER for each frame is the 2D position of the deformable object, the weights w of the training images and the approximate appearance $\hat{a}$ of the hidden state. The AHM-TRACKER can be extended to output other information, such as a decision about the state of the deformable object. This requires additional information about the training set. For example, to accurately detect facial expressions, each training image can be annotated with a label "smiling," "frowning," "neutral," etc. To output such additional information, the AHM-TRACKER can use the weights w' and these labels to recognize (classify) the current object transformation, for example, facial expression.

3. Experiments and Results

We implemented the AHM-TRACKER in C++ on an Intel Dual 2.10 GHz CPU. The AHM-TRACKER runs in real time. Both training and testing videos were recorded using a Dell XPS M1330 Webcam.

The AHM tracker was tested on three datasets which we call (1) Head Pose (2) Facial Expressions and (3) Deformable Candy. We established the ground truth labels of the training data manually. These labels classify the specific facial expressions, head orientations or deformation states of the candy wrapper. Our training data did not need any explicit manual feature marking. The datasets 1 and 2 are representative of input data to human-computer interfaces, for example, a camera-based mouse-replacement system for people with disabilities [2]. Dataset 3 has been included to demonstrate the breadth of applications of this system.

The AHM-TRACKER uses the threshold kernel with grayscale threshold τ. The AHM-TRACKER uses the APPEARANCE APPROXIMATION algorithm and a local positional search with a binary alignment operator (Hamming distance). Each video sequence is tracked using a training set with pre-computed Gram matrices.

To compensate for fast motion of the object of interest, we applied the AHM-TRACKER on each video frame four times, each time incrementally updating the position estimate p and $\hat{a}$.

3.1. Head Pose Estimation

Dataset 1 involved four subjects – two men, two women. We collected 41–48 training images of size 80 by 100 pixels, consisting the lower head region at different viewing directions (Fig. 3). From each training set, we surprisingly only needed a dynamic subset size of 5 for all head pose experiments.

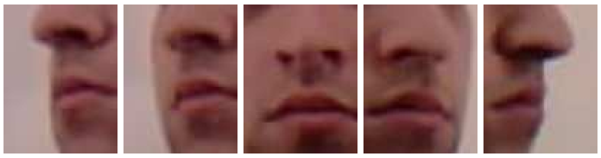

Figure 3. Sample Training Images of Subject B in Head Pose Experiment.

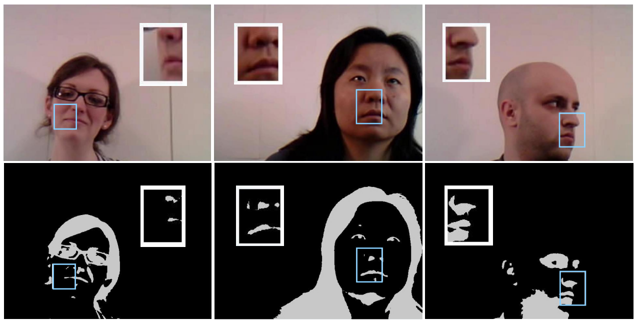

Figure 4. Tracked movements of faces of subjects A, B, and D. Top: Original frames with detected objects (blue bounding boxes) and best matching training images q^* (inset). Bottom: Binary images computed from threshold kernel k and weights w with detected objects z (blue bounding box) at location p and reconstructed template $\hat{a}$ (inset) with similar appearance of z. Correction detections are shown for subjects B and D, and a misalignment for subject A.

We instructed the subjects to turn their heads 180-degrees and move rapidly. The videos we recorded of these movements contained 382 frames on average. Frame-by-frame inspection of the video verified that a full spectrum of different face poses was recorded for each subject. These motions are more general than the motions captured by standard ASM and AAM formulations, due to the 2D nature of their models (points/polygons). AAM-based methods re-

Table 1. Results of head pose experiment: The detection rates for the four accuracy measures show that the various head poses of subjects B–D were recognized accurately (see "Correct" detections in (1)). The mis-detections for subject A were due to the distance of the subject to the camera and the low contrast recording of her face.

Video			Accuracy Measures (%)			
Subj.	# Frames	Param.	(1)	(2)	(3)	(4)
A-1	294	τ_1–τ_4	39	31	29	1
A-2	275	τ_1–τ_4	42	0	45	13
B-1	427	τ_2–τ_4	100	0	0	0
B-2	472	τ_2–τ_4	96	4	0	0
B-3	344	τ_2–τ_4	98	1	1	0
B-4	344	τ_2–τ_4	100	0	0	0
C-1	772	τ_2–τ_3	87	8	5	0
C-2	351	τ_2–τ_3	76	16	8	0
C-3	344	τ_2–τ_3	96	4	0	0
C-4	169	τ_2–τ_3	100	0	0	0
D-1	494	τ_1–τ_4	100	0	0	0
D-2	294	τ_1–τ_4	89	9	2	0

quire non-trivial formulations [7] to recover from the kind of self-occlusions of the face that we included in our experiments.

Since it was not immediately apparent which threshold to use for the binary kernel, we tested four different thresholds τ_1–τ_4 (grey-levels 30, 50, 70, and 90). We found that the extreme thresholds τ_1 or τ_4 could not be used for some subjects due to their skin color (column 3 in Table 1).

The accuracy of the position and orientation estimates was determined by using manual marking over all frames. To measure accuracy of the orientation estimate of the AHM-TRACKER, the current frame was compared to the training image with the highest weight, a very conservative measure of the orientation output. The measures of accuracy are: (1) *Correct* - the AHM-TRACKER exactly estimates position and orientation, (2) *Unknown Orientation* - the AHM-TRACKER is correctly aligned, but the max-weighted training image does not provide the correct alignment, (3) *Misaligned* - the AHM-Tracker is tracking the object, but there is a misalignment of features (e.g., the left nostril is matched with the right nostril), (4) *Lost* - the AHM-TRACKER is not tracking the object of interest.

The detection rates for the four accuracy measures show that the estimates of the various head orientations and positions of subjects B–D were extremely accurate, with correct performance close to 100 % (Table 1). Subject A was the most difficult target to track, due to her distance from the camera, and the low contrast recording of her face.

3.2. Facial Expression

We tested the AHM-TRACKER on video of extreme changes in facial expressions. The training set consists of 19 training images of dimension 183×107 of a mouth smiling, neutral, and frowning (Fig. 5). The purpose of this experiment is to show the completeness of the AHM-Tracker and also the effects of the subset size on accuracy. The AHM-Tracker was run on the same test video using dynamic subset sizes of 1 through 4. This video contains 444 frames of a subject making extreme facial expressions. The accuracy was determined by a manual comparison of the video to the highest weighted training image. Our results show that the three expressions can be detected accurately (Fig. 5).

The recognition rates using subset sizes of 3 and 4 were high, with success rates of 95% and 96%, respectively (Table 2). Remarkably, the recognition accuracy rate was still reasonably high using a subset size of 2, with a success rate of 82%. The trivial case of a subset size of 1 yields a success rate of 66%.

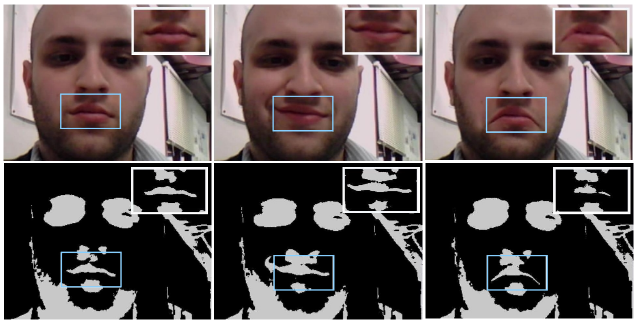

Figure 5. Tracked deformations of facial expressions (explanation in Fig. 4). The expressions of the best-matching template (insets) accurately represent the detected object of interest, here the mouth.

Table 2. Results of facial expression experiments as a function of the dynamic subset size. The three measures were computed by averaging the detection performance over the 444 frames of the facial-motion test video.

Subset Size	Correct (%)	Marginal (%)	Incorrect (%)
1	66	14	20
2	82	7	11
3	95	1	4
4	96	2	2

3.3. Candy Deformation

The AHM-TRACKER was tested on four videos of a candy wrapper being squeezed and moved (228–401 frames). The training set consists of 38 training images of dimension 186×65. We used a dynamic subset size of 8. Our experiment showed that our method tracked the position and deformation of the wrapper correctly (see sample results in Fig. 6).

3.4. Competing Method

The first application of AHM-TRACKER will be to replace a very popular, worldwide-used, template-based HCI

Figure 6. Tracked deformations of a candy wrapper (explanation in Fig. 4). The tracker correctly detects the increasing level of folding.

system [2]. This system was tested on subject B in 50 sequences performing the same extreme motions as the head pose and facial expression experiments, with an unacceptable rate of 9 lost features out of 50 trials. In comparison, our approach did not result in tracking failure in almost all cases and successfully recovered from misalignments.

4. Prediction Theory of Active Hidden Models

The parameter τ of the threshold kernel was determined experimentally. This introduces an open problem regarding Active Hidden Models:

Given a representative training set, how does the choice of a kernel effect the accuracy of the AHM-TRACKER*?*

In this section we delve into some of the theoretic aspects of Active Hidden Models and give a partial answer to this question. To address the notion of "accuracy," we first assume that the real-time observation z is the realization of a random variable $\mathcal{Z}$ over observation space Z. This implies that its corresponding hidden state x is also the realization of a hidden random variable.

The random variable $\mathcal{Z}$ represents a measure of the observations (and hidden states) of the AHM-TRACKER. We prove there exists an unbiased estimate of the covariance of the kernel projection for a special set of kernels where the training set Q is centered. A set Q is centered with respect to a kernel k if

$$\sum_{i=1}^{n} \phi(q_i) \quad = \quad 0. \tag{8}$$

From this definition, we can now use the structure of the Gram matrix of centered kernels to predict the accuracy of their kernel projections.

Theorem 1. *Given is the realization $z \in Z$ of random variable $\mathcal{Z}$. If the set Q is centered with respect to k and consists of independent samples of $\mathcal{Z}$, then an unbiased estimate $\hat{C}$ of the covariance of the kernel projection of z (Eq. 1) is of the form:*

$$\hat{C} = \frac{1}{n} K^{+} K$$

The proof can be found in the appendix.

Corollary 1. *The trace of the projection covariance estimate $\hat{C}$ is equal to the rank of K:*

$$\mathit{trace}(\hat{C}) = \mathit{rank}(K).$$

This corollary follows directly from the definition of $K^{+}K$ and says that the covariance of the kernel projection is directly proportional the level of linear independence of the mapping of training set Q in the state space. These results have not been confirmed experimentally, since the scope of this paper is limited to threshold kernels, whose parameter τ only effected the rank of the Gram matrices in the extreme ends of its range. Another open issue is the extension of Theorem 1 to uncentered kernels.

Our theoretical results give a general intuition about a symmetry between classification and tracking. In classification, kernels are used to make training data linearly separable. We show the optimal (minimum variance) tracking kernels are those that make the training observations linearly dependent.

5. Discussion and Conclusions

We tested our AHM-TRACKER to determine head pose, facial expressions, and the deformations of nonrigid objects. The head pose experiments showed that our AHM-TRACKER was very successful in determining extreme head orientations. The facial expression experiments showed that the AHM-TRACKER can accurately predict states of the deformable object with dynamic subsets of size as small as 2. The inclusion of the candy wrapper experiment showed that the AHM-TRACKER has a broad range of applications.

The AHM-TRACKER requires user-specific training sets for the head pose experiments due to the properties of the threshold kernel. This is not a serious restriction for human-computer interface settings that require head position and facial expression recovery because user-specific training sets can be saved across sessions. The AHM-TRACKER can be immediately applied to improve camera-based mouse-replacement systems for people with disabilities. In future work, we will explore kernels that allow a single Active Hidden Model to track multiple users.

A key benefit of the AHM-Tracker is that it runs in real time with frequencies up to four times per frame. The tracker is easy to set up because AHMs are completely defined by a set of unmarked training images and a kernel. The only pre-computing necessary is the Gram matrix and its (pseudo)inverse.

The limitations of the AHM-TRACKER are closely tied to the choice of kernel used. The AHMs we tested are based on the threshold kernel, which forces the training images to have an illumination that is similar to the real-time object. In future work, we will address this issue by experimenting with lighting invariant kernels and incremental learning [11].

We showed that the optimal tracking kernels are those that make the training observations linearly dependent (Sec. 4). Linear dependence of the training set generally implies greater difficulty in inferring information (deciding) about the current object of interest, as discussed at the end of Sec. 2.4). This result indicates a trade-off in the choice of kernels. A higher rank of the Gram matrix of a kernel results in greater expressiveness of decision boundaries but more variance in the kernel projections. Future work will aim to confirm the results of theorem 1 experimentally.

Appendix: Kernel Projection Covariance

Theorem 1. *Given is the realization $z \in Z$ of random variable $\mathcal{Z}$. If the set Q is centered with respect to k and consists of independent samples of $\mathcal{Z}$, then an unbiased estimate $\hat{C}$ of the covariance of the kernel projection of z (Eq. 1) is of the form:*

$$\hat{C} = \frac{1}{n} K^{+} K$$

Proof. Using standard covariance properties and the limit definition of pseudoinverses, the covariance of the kernel projection is of the form:

$$C = \lim_{\delta \to 0} (K + \delta I)^{-1} C_k (K + \delta I)^{-1}$$

The $n \times n$ matrix C_k represents the covariance of the empirical map $k(z)$. Since Q is a representative sample of $\mathcal{Z}$, an unbiased estimate of C_k is the sample covariance matrix

$$\hat{C}_k = \frac{1}{n} \sum_{i=1}^{n} (k(q_i) - k(Q)) (k(q_i) - k(Q))^T .$$

The mean $k(Q) = \frac{1}{n} \sum_{i=1}^{n} k(q_i)$ is equal to 0, given the centering assumption (Eq. 8). The covariance of the empirical map C_k can therefore be simplified to

$$\hat{C}_k = \frac{1}{n} \sum_{i=1}^{n} k(q_i) k(q_i)^T = \frac{1}{n} K^T K.$$

A more general definition of the Moore Penrose pseudoinverse is needed to find the covariance estimate $\hat{C}$. Using Tikhonov regularization, it can be shown that the Moore-Penrose pseudoinverse is of the form:

$$A^{+} = \lim_{\delta \to 0} \left(A^T A + \delta\, \Gamma^T \Gamma \right)^{-1} A^T, \qquad (9)$$

where the term Γ represents an $n \times n$ matrix. Using this definition, we can derive the final form of $\hat{C}$. Relative positions of K can be exchanged since $K = K^T$ is symmetric.

$$\begin{aligned} \hat{C} &= \lim_{\delta \to 0} (K + \delta I)^{-1} \hat{C}_k (K + \delta I)^{-1} \\ &= \frac{1}{n} \lim_{\delta \to 0} \left(K^T K + \delta (2K + \delta I) \right)^{-1} K^T K \end{aligned}$$

Since K is symmetric and positive semidefinite, the term $(2K + \delta I)$ is symmetric and positive definite, and therefore admits a Cholesky decomposition into $\Gamma^T \Gamma$. Thus, by Eq. 9:

$$\hat{C} = \frac{1}{n} K^{+} K.$$

This estimate is unbiased since it is the linear transformation of $\hat{C}_k$, which is itself an unbiased estimator. □

References

[1] S. Avidan. Support vector tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(8):1064–1072, 2004.

[2] M. Betke, J. Gips, and P. Fleming. The camera mouse: Visual tracking of body features to provide computer access for people with severe disabilities. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 10:1–10, 2002.

[3] M. J. Black and A. D. Jepson. Eigentracking: Robust matching and tracking of articulated objects using a view-based representation. *International Journal of Computer Vision*, 26(1):63–84, 2004.

[4] D. Comaniciu, V. Ramesh, and P. Meer. Kernel-based object tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25:564–575, 2003.

[5] T. F. Cootes, G. J. Edwards, and C. J. Taylor. Active appearance models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(6):681–685, 2001.

[6] T. F. Cootes, C. J. Taylor, D. H. Cooper, and J. Graham. Active shape models – their training and application. *Computer Vision and Image Understanding*, 61(1):38–59, 1995.

[7] R. Gross, I. Matthews, and S. Baker. Constructing and fitting active appearance models with occlusion. *IEEE Conference on Computer Vision and Pattern Recognition Workshop*, pages 72–72, June 2004.

[8] B. Han, Y. Zhu, D. Comaniciu, and L. Davis. Kernel-based Bayesian filtering for object tracking. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 227–234, 2005.

[9] F. D. la Torre Frade and M. H. Nguyen. Parameterized kernel principal component analysis: Theory and applications to supervised and unsupervised image alignment. *IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[10] M. Dewan and G. D. Hager. Toward optimal kernel-based tracking. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 618–625, 2006.

[11] D. A. Ross, J. Lim, R.-S. Lin, and M.-H. Yang. Incremental learning for robust visual tracking. *International Journal of Computer Vision*, 77(1–3):125–141, 2008.

[12] B. Schölkopf, S. Mika, C. J. C. Burges, P. Knirsch, K.-R. Müller, G. Rätsch, and A. J. Smola. Input space versus feature space in kernel-based methods. *IEEE Transactions on Neural Networks*, 10(5):1000–1017, 1999.

[13] B. Schölkopf and A. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. The MIT Press, 2001.

[14] S. Sclaroff and J. Isidoro. Active blobs. *IEEE International Conference on Computer Vision*, page 1146, 1998.

[15] C. J. Twining and C. J. Taylor. Kernel principal component analysis and the construction of non-linear active shape models. *British Machine Vision Conference*, 2001.

[16] P. Viola, M. J. Jones, and D. Snow. Detecting pedestrians using patterns of motion and appearance. *International Journal of Computer Vision*, 63(2):153–161, 2005.

[17] Y. Zhong, A. K. Jain, and M.-P. Dubuisson-Jolly. Object tracking using deformable templates. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(5):544–549, 2000.