The Template Update Problem

Iain Matthews, Takahiro Ishikawa, and Simon Baker
The Robotics Institute, Carnegie Mellon University
5000 Forbes Avenue, Pittsburgh, PA 15213, USA
{iainm,taka,simonb}@cs.cmu.edu

Abstract

Template tracking is a well studied problem in computer vision which dates back to the Lucas-Kanade algorithm of 1981. Since then the paradigm has been extended in a variety of ways including: arbitrary parametric transformations of the template, and linear appearance variation. These extensions have been combined, culminating in non-rigid appearance models such as Active Appearance Models (AAMs) and Active Blobs. One question that has received very little attention is how to update the template over time so that it remains a good model of the object being tracked. This paper proposes an algorithm to update the template that avoids the "drifting" problem of the naive update algorithm. Our algorithm can be interpreted as a heuristic to avoid local minima. It can also be extended to templates with linear appearance variation. This extension can be used to convert (update) a generic, person-independent AAM into a person specific AAM.

1 Introduction

Template tracking is a well studied problem in computer vision which dates back to [7]. An object is tracked through a video sequence by extracting an example image of the object, a *template*, in the first frame and then finding the region which matches the template as closely as possible in the remaining frames. Template tracking has been extended in a variety of ways, including: (1) to allow arbitrary parametric transformations of the template [3], (2) to allow linear appearance variation [4, 6], and (3) to be efficient [6, 2]. Combining these extensions has resulted in non-rigid appearance models such as Active Appearance Models (AAMs) [5] and Active Blobs [9].

The underlying assumption behind template tracking is that the appearance of the object remains the same throughout the entire video. This assumption is generally reasonable for a certain period of time, but eventually the template is no-longer an accurate model of the appearance of the object. A naive solution to this problem is to update the template every frame (or every n frames) with a new template extracted from the current image at the current location of the template. The problem with this naive algorithm is that the template "drifts." Each time the template is updated, small errors are introduced in the location of the template. With each update, these errors accumulate and the template steadily drifts away from the object. See Figure 1 for an example.

In this paper we propose a template update algorithm that does not suffer from drift. The template can be updated in every frame and yet still stays firmly attached to the original object. The algorithm is a simple extension of the naive algorithm. As well as

maintaining a current estimate of the template, our algorithm also retains the first template from the first frame. The template is first updated as in the naive algorithm with the image at the current template location. However, to eliminate drift, this updated template is then aligned with the first template to give the final update. We first evaluate this algorithm *qualitatively* and show that it can update the template without introducing drift. Next, we reinterpret the algorithm as a heuristic to avoid local minima and *quantitatively* evaluate it as such.

We then consider the more general case of template tracking with linear appearance variation. Specifically we generalize our template update algorithm to AAMs [5]. In this context our appearance update algorithm can also be interpreted as a heuristic to avoid local minima and so we again quantitatively evaluate it as such. We also demonstrate how our algorithm can be applied to automatically convert a generic person-independent AAM into a person specific AAM.

2 Template Tracking

We begin by considering the original template tracking problem where the object is represented by a single template image. Suppose we are given a video sequence of images $I_n(\mathbf{x})$ where $\mathbf{x} = (x,y)^{\mathrm{T}}$ are the pixel coordinates and $n = 0,1,2,\ldots$ is the frame number. In template tracking, a subregion of the initial frame $I_0(\mathbf{x})$ that contains the object of interest is extracted and becomes the template $T(\mathbf{x})$. (The template is not necessarily rectangular, and might, for example, be a face shaped region [5].)

Let $\mathbf{W}(\mathbf{x};\mathbf{p})$ denote the parameterized set of allowed deformations of the template, where $\mathbf{p} = (p_1,\ldots p_k)^{\mathrm{T}}$ is a vector of parameters. The warp $\mathbf{W}(\mathbf{x};\mathbf{p})$ takes the pixel $\mathbf{x}$ in the coordinate frame of the template $T(\mathbf{x})$ and maps it to a sub-pixel location $\mathbf{W}(\mathbf{x};\mathbf{p})$ in the coordinate frame of the video $I_n(\mathbf{x})$. The set of allowed warps depends on the type of motions we expect from the object being tracked. If the object is a roughly planar image patch moving in 2D we might consider the set of *similarity warps*:

$$\mathbf{W}(\mathbf{x};\mathbf{p}) = \begin{pmatrix} (1+p_1)\cdot x & - & p_2\cdot y & + & p_3 \\ p_2\cdot x & + & (1+p_1)\cdot y & + & p_4 \end{pmatrix} \tag{1}$$

where there are 4 parameters $\mathbf{p} = (p_1,p_2,p_3,p_4)^{\mathrm{T}}$. In general, the number of parameters k may be arbitrarily large and $\mathbf{W}(\mathbf{x};\mathbf{p})$ can be arbitrarily complex. (A complex example is the set of piecewise affine warps used to model non-rigidly moving objects in Active Appearance Models [5].)

The goal of template tracking is to find the best match to the template in every subsequent frame in the video. The sum of squared error is normally used to measure the degree of match between the template and the video frames. The goal is therefore to compute:

$$\mathbf{p}_n \;=\; \arg\min_{\mathbf{p}} \sum_{\mathbf{x}\in T} [I_n(\mathbf{W}(\mathbf{x};\mathbf{p})) - T(\mathbf{x})]^2 \tag{2}$$

for $n \geq 1$ and where the summation is over all of the pixels in the template. (Excuse the abuse of terminology.) The original solution to the non-linear optimization in Equation (2) was the Lucas-Kanade algorithm [7]. A variety of other algorithms have since been proposed. See [2] for a recent survey.

2.1 Template Update Strategies

In this paper we consider the problem of how to update the template $T(\mathbf{x})$. Suppose that a (potentially) different template is used in each frame. Denote the template that is used in the n^{th} frame $T_n(\mathbf{x})$. Tracking then consists of computing:

$$\mathbf{p}_n \;=\; \arg\min_{\mathbf{p}} \sum_{\mathbf{x}\in T_n} \left[I_n(\mathbf{W}(\mathbf{x};\mathbf{p})) - T_n(\mathbf{x}) \right]^2 \tag{3}$$

and the template update problem consists of computing $T_{n+1}(\mathbf{x})$ from $I_0(\mathbf{x}),\ldots,I_n(\mathbf{x})$ and $T_1(\mathbf{x}),\ldots,T_n(\mathbf{x})$. The simplest strategy is not to update the template at all:

Strategy 1: No Update

$$T_{n+1}(\mathbf{x}) \;=\; T_1(\mathbf{x}) \text{ for all } n \geq 1.$$

The simplest strategy for actually updating the template is to set the new template to be the region of the input image that the template was tracked to in $I_n(\mathbf{x})$:

Strategy 2: Naive Update

$$T_{n+1}(\mathbf{x}) \;=\; I_n(\mathbf{W}(\mathbf{x};\mathbf{p}_n)) \text{ for all } n \geq 1.$$

Neither of these strategies are very good. With the first strategy, the template eventually, and inevitably, becomes out-of-date and no longer representative of the appearance of the object being tracked. With the second strategy, the template eventually drifts away from the object. Small errors in the warp parameters $\mathbf{p}_n$ mean that the new template $I_n(\mathbf{W}(\mathbf{x};\mathbf{p}_n))$ is always a slighted shifted version of what it ideally should be. These errors accumulate and after a while the template drifts away from the object that it was initialized to track. See Figure 1 for an example of the template drifting in this way. (Note that simple variants of this strategy such as updating the template every few frames, although more robust, also suffer from the same drifting problem.)

How can we update the template every frame and avoid it wandering off? One possibility is to keep the first template $T_1(\mathbf{x})$ around and use it to correct the drift in $T_{n+1}(\mathbf{x})$. For example, we could take the estimate of $T_{n+1}(\mathbf{x})$ computed in Strategy 2 and then align $T_{n+1}(\mathbf{x})$ to $T_1(\mathbf{x})$ to eliminate the drift. Since $T_{n+1}(\mathbf{x}) = I_n(\mathbf{W}(\mathbf{x};\mathbf{p}_n))$ this is the same as first tracking in image $I_n(\mathbf{x})$ with template $T_n(\mathbf{x})$ and then with template $T_1(\mathbf{x})$. If the non-linear minimizations in Equations (2) and (3) are solved perfectly, this is theoretically exactly the same as just tracking with $T_1(\mathbf{x})$. The non-linear minimizations are solved using a gradient descent algorithm, however, and so this strategy is actually different. Let us change the notation slightly to emphasize the point that a gradient descent algorithm is used to solve Equation (3). In particular, re-write Equation (3) as:

$$\mathbf{p}_n \;=\; \text{gd}\min_{\mathbf{p}=\mathbf{p}_{n-1}} \sum_{\mathbf{x}\in T_n} \left[I_n(\mathbf{W}(\mathbf{x};\mathbf{p})) - T_n(\mathbf{x}) \right]^2 \tag{4}$$

where $\text{gd}\min_{\mathbf{p}_{n-1}}$ means "perform a gradient descent minimization" starting at $\mathbf{p}=\mathbf{p}_{n-1}$. To correct the drift in Strategy 2, we therefore propose to compute updated parameters:

$$\mathbf{p}_n^* \;=\; \text{gd}\min_{\mathbf{p}=\mathbf{p}_n} \sum_{\mathbf{x}\in T_1} \left[I_n(\mathbf{W}(\mathbf{x};\mathbf{p})) - T_1(\mathbf{x}) \right]^2. \tag{5}$$

Note that this is different from tracking with the constant template $T_n = T_1$ using:

$$\text{gd} \min_{\mathbf{p}=\mathbf{p}_{n-1}} \sum_{\mathbf{x}\in T_1} [I_n(\mathbf{W}(\mathbf{x};\mathbf{p})) - T_1(\mathbf{x})]^2 \tag{6}$$

because the starting point of the gradient descent is different. To correct the drift, we use $\mathbf{p}_n^*$ rather than $\mathbf{p}_n$ to form the template for the next image. In summary (see also Figure 2), we update the template using:

Strategy 3: Template Update with Drift Correction

If $\|\mathbf{p}_n^* - \mathbf{p}_n\| \leq \varepsilon$ then $T_{n+1}(\mathbf{x}) = I_n(\mathbf{W}(\mathbf{x};\mathbf{p}_n^*))$

else $T_{n+1}(\mathbf{x}) = T_n(\mathbf{x})$

where $\varepsilon > 0$ is a small threshold that enforces the requirement that the result of the second gradient descent does not diverge too far from the result of the first. If it does, there must be a problem and so we act conservatively by not updating the template in that step. (A minor variant of this is to perform the drift-correcting alignment using the magnitudes of the gradients of the image and the template rather than the raw images to increase robustness to illumination variation.)

2.2 Qualitative Comparison

We now present a *qualitative* comparison of the three update strategies. Although we only have room to include one set of results, these results are typical. A more principled *quantitative* evaluation is included in Section 2.4. We implemented each of the three strategies and ran them on a 972 frame video of a car tracked using a 2D similarity transform.

Sample frames are shown in Figure 1 for each of the update algorithms. If the template is not updated (Strategy 1), the car is no longer tracked correctly after frame 312. If we update the template every frame using the naive approach (Strategy 2), by around frame 200 the template has drifted away from the car. With update Strategy 3 "Template Update with Drift Correction", the car is tracked throughout the entire sequence and the template is updated correctly in every frame, without introducing any drift. See the accompanying movie[1] "`car-track.mpg`" for tracking results on the sequence.

2.3 Reinterpretation of Update Strategy 3

A schematic diagram of Strategy 3 is included in Figure 2(a). The image $I_n(\mathbf{x})$ is first tracked with template $T_n(\mathbf{x})$ starting from the previous parameters $\mathbf{p}_{n-1}$. The result is the tracked image $I_n(\mathbf{W}(\mathbf{x};\mathbf{p}_n))$ and the parameters $\mathbf{p}_n$. The new template $T_{n+1}(\mathbf{x}) = I_n(\mathbf{W}(\mathbf{x};\mathbf{p}_n^*))$ is then computed by tracking $T_1(\mathbf{x})$ in $I_n(\mathbf{x})$ starting at parameters $\mathbf{p}_n$.

If we reorganize Figure 2(a) slightly we get Figure 2(b). The only change made in this reorganization is that the "tracked output" is $I_n(\mathbf{W}(\mathbf{x};\mathbf{p}_n^*))$ rather than $I_n(\mathbf{W}(\mathbf{x};\mathbf{p}_n))$. The difference between Figure 2(a) and Figure 2(b) is not the computation (the two diagrams result in the same sequence of parameters $\mathbf{p}_n$), but their interpretation. Figure 2(a) can be interpreted as tracking with $T_n(\mathbf{x})$ followed by updating $T_n(\mathbf{x})$. Figure 2(b) can be interpreted as tracking with $T_n(\mathbf{x})$ to get an initial estimate to track with $T_1(\mathbf{x})$. This initial estimate improves robustness because tracking with $T_1(\mathbf{x})$ is prone to local minima.

[1] Movies may be downloaded from `http://www.ri.cmu.edu/projects/project_513.html`.

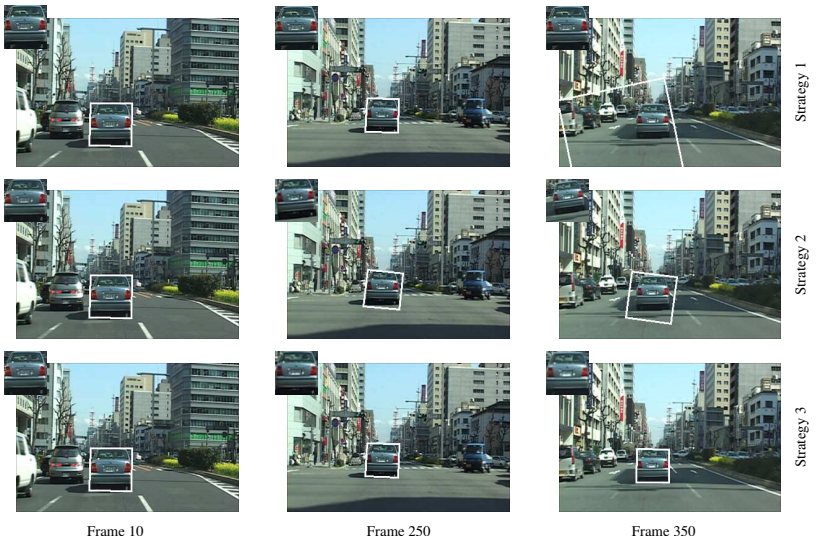


Figure 1: A qualitative comparison of update Strategies 1, 2, and 3. With Strategy 1 the template is not updated and tracking eventually fails. With Strategy 2, the template is updated every frame and the template "drifts". With Strategy 3 the template is updated every frame, but a "drift correction" step is added. With this strategy the object is tracked correctly.

Tracking with $I_{n-1}(\mathbf{W}(\mathbf{x};\mathbf{p}^*_{n-1}))$ is less prone to local minima and is used to initialize the tracking with $T_1(\mathbf{x})$ and start it close enough to avoid local minima. In summary, there are two equivalent ways to interpret Strategy 3:

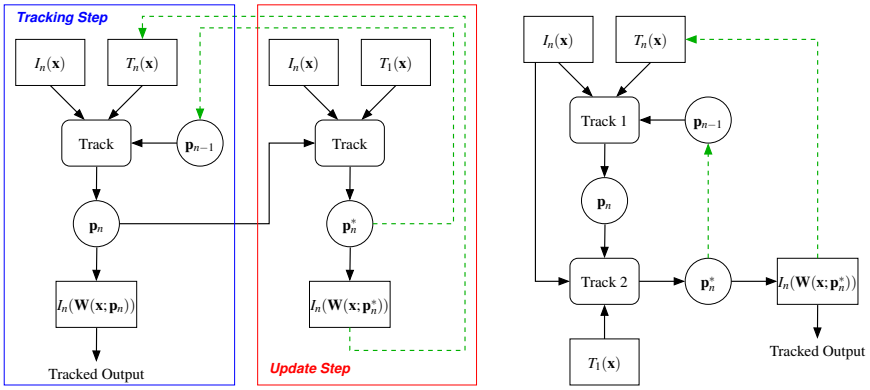
1. The template can be updated every frame, but it must be re-aligned to the original template $T_1(\mathbf{x})$ to remove drift.

2. Not updating the template and tracking using the constant template $T_1(\mathbf{x})$ is fine, so long as we first initialize $\mathbf{p}_n$ by tracking with $T_n(\mathbf{x}) = I_{n-1}(\mathbf{W}(\mathbf{x};\mathbf{p}^*_{n-1}))$.

2.4 Quantitative Evaluation

We now present a *quantitative* evaluation of Strategy 3 in the context of the second interpretation above. We measure how much more robust tracking is if we initialize it by first tracking with $I_{n-1}(\mathbf{W}(\mathbf{x};\mathbf{p}^*_{n-1}))$; i.e. use Strategy 3 rather than Strategy 1.

Our goal is to track the car in the 972 frame video sequence shown in Figure 1. First, using a combination of Lucas-Kanade tracking and hand re-initialization, we obtain a set of ground-truth parameters $\mathbf{p}_n$ for each frame. We then generate 50 test cases for each of the 972 frames by randomly perturbing the ground-truth parameters $\mathbf{p}_n$. The perturbation is computed using a normal distribution so that the root-mean-square template coordinate locations in the image are displaced by a known spatial standard deviation. We then run the two tracking algorithms starting with the same perturbed parameters and determine which of the two algorithms converged by comparing the final $\mathbf{p}_n$ with the ground-truth. This experiment is repeated for all frames over a range of perturbation standard deviations. The final result is a graph plotting the frequency of convergence versus the perturbation magnitude for each algorithm. The results of this comparison are shown in Figure 3. We



(a) Update Strategy 3 (b) Update Strategy 3 (Reorganized)

Figure 2: Two equivalent schematic diagrams for update Strategy 3. The diagrams are equivalent in the sense that they result in exactly the same sequence of parameters $\mathbf{p}_n$. (a) Can be interpreted as first tracking with template T_n and then updating T_n. (b) Can be interpreted as tracking with constant template T_1, after first tracking with $T_n(\mathbf{x}) = I_{n-1}(\mathbf{W}(\mathbf{x};\mathbf{p}_{n-1}^*))$ to avoid local minima.

plot two curves, one for update Strategy 1 "No Update" and one for update Strategy 3 "Template Update with Drift Correction". No results are shown for Strategy 2 because after a few frames the template drifts and so none of the trials converge to the correct location (although many trials do converge). The accompanying movie "`car-exp.mpg`" shows example trials for both algorithms with the ground truth marked in yellow and the perturbed position tracked in green. Figure 3 clearly demonstrates that updating the template using Strategy 3 dramatically improves the tracking robustness.

3 Template Tracking With Appearance Variation

We now consider the problem of template tracking with linear appearance variation. Instead of tracking with a single template $T_n(\mathbf{x})$ (for each frame n), we assume that a linear model of appearance variation is used; i.e. a set of appearance images $A_n^i(\mathbf{x})$ where $i = 1,\ldots,d_n$. Instead of the template $T_n(\mathbf{x})$ appearing (appropriately warped) in the input image $I_n(\mathbf{x})$, we assume that:

$$T_n(\mathbf{x}) + \sum_{i=1}^{d_n} \lambda^i A_n^i(\mathbf{x}) \tag{7}$$

appears instead for a unknown set of *appearance parameters* $\boldsymbol{\lambda} = (\lambda^1,\ldots,\lambda^{d_n})^{\mathrm{T}}$. The appearance images $A_n^i(\mathbf{x})$ can be used to model either illumination variation [6] or more general linear appearance variation [4, 5]. In this paper, we focus particularly on Active Appearance Models [5, 8] which combine a linear appearance model with a (low parametric) piecewise affine warp to model the shape deformation $\mathbf{W}(\mathbf{x};\mathbf{p})$. The process of tracking with such a linear appearance model then consists of minimizing:

$$(\mathbf{p}_n,\boldsymbol{\lambda}_n) \;=\; \arg\min_{(\mathbf{p},\boldsymbol{\lambda})} \sum_{\mathbf{x}\in T_n} \left[I_n(\mathbf{W}(\mathbf{x};\mathbf{p})) - T_n(\mathbf{x}) - \sum_{i=1}^{d_n} \lambda^i A_n^i(\mathbf{x}) \right]^2 . \tag{8}$$

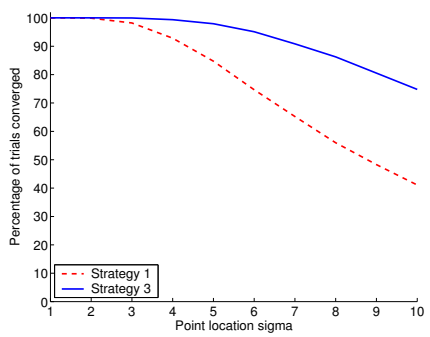


Figure 3: The frequency of convergence of Strategies 1 and 3 plot against the magnitude of the perturbation to the ground-truth parameters, computed over 50 trials for each frame in the sequence used in Figure 1. The results demonstrate that updating the template using Strategy 3 results in far more robust tracking.

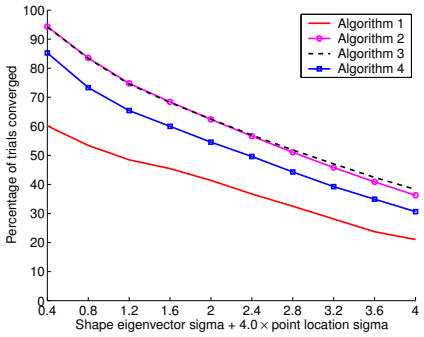


Figure 4: A comparison of the frequency of convergence of four template and appearance model update algorithms. The three algorithms which actually update the template and/or appearance model (Algorithms 2, 3, and 4) all dramatically outperform the algorithm which does not update model (Algorithm 1).

Several efficient gradient descent algorithms have been proposed to solve this non-linear optimization problem including [6] for translations, affine warps, and 2D similarity transformations, [1] for arbitrary warps that form a group, and [8] for Active Appearance Models. Denote the result:

$$(\mathbf{p}_n, \boldsymbol{\lambda}_n) = \text{gd} \min_{(\mathbf{p}_{n-1}, \boldsymbol{\lambda}_{n-1})} \sum_{\mathbf{x} \in T_n} \left[I_n(\mathbf{W}(\mathbf{x};\mathbf{p})) - T_n(\mathbf{x}) - \sum_{i=1}^{d_n} \lambda^i A_n^i(\mathbf{x}) \right]^2 \tag{9}$$

where the gradient descent is started at $(\mathbf{p}_{n-1}, \boldsymbol{\lambda}_{n-1})$.

3.1 Updating Both the Template and the Appearance Model

Assume that the initial template T_1 and appearance model A_1^i are given. The template update problem with linear appearance variation then consists of estimating T_{n+1} and A_{n+1}^i from $I_0(\mathbf{x}), \ldots, I_n(\mathbf{x})$, $T_1(\mathbf{x}), \ldots, T_n(\mathbf{x})$, and $A_1^i, \ldots, A_n^i$. Analogously to above, denote:

$$(\mathbf{p}_n^*, \boldsymbol{\lambda}_n^*) = \text{gd} \min_{(\mathbf{p}_n, \boldsymbol{\lambda}_n)} \sum_{\mathbf{x} \in T_n} \left[I_n(\mathbf{W}(\mathbf{x};\mathbf{p})) - T_1(\mathbf{x}) - \sum_{i=1}^{d_1} \lambda^i A_1^i(\mathbf{x}) \right]^2 . \tag{10}$$

One way to update the template and appearance model is then as follows:

Strategy 4: Template and Appearance Model Update with Drift Correction

If $\|\mathbf{p}_n^* - \mathbf{p}_n\| \leq \varepsilon$ then $(T_{n+1}(\mathbf{x}), A_{n+1}^i) = \text{PCA}(I_1(\mathbf{W}(\mathbf{x};\mathbf{p}_1^*)), \ldots, I_n(\mathbf{W}(\mathbf{x};\mathbf{p}_n^*)))$

else $T_{n+1}(\mathbf{x}) = T_n(\mathbf{x})$, $A_{n+1}^i = A_n^i$

where PCA() means perform Principal Components Analysis setting T_n to be the mean and A_n^i to be the first d_n eigenvectors, where d_n is chosen to keep a fixed amount of the energy, typically 95%. (Other variants of this exist, such as incrementally updating

appearance model A_n^i to include the new measurement $I_n(\mathbf{W}(\mathbf{x};\mathbf{p}_n^*))$.) If we reinterpret this algorithm as in Section 2.3, we end up with the following two step tracking algorithm:

Step 1: Apply PCA to $I_1(\mathbf{W}(\mathbf{x};\mathbf{p}_1^*)),\ldots,I_{n-1}(\mathbf{W}(\mathbf{x};\mathbf{p}_{n-1}^*))$. Set T_n to be the mean vector and A_n^i to be the first $i=1,\ldots,d_n$ eigenvectors. Once computed, track with template T_n and appearance model A_n^i.

Step 2: Track with the *a priori* template $T_1(\mathbf{x})$ and linear appearance model $A_1^i(\mathbf{x})$, starting the gradient descent at the result of the first step.

One way to interpret these two steps is as performing "progressive appearance complexity", analogously to "progressive transformation complexity" [3] the standard heuristic for improving the robustness of tracking algorithms by increasing the complexity of the warp $\mathbf{W}(\mathbf{x};\mathbf{p})$. For example, tracking with an affine warp is often performed by first tracking with a translation, then a 2D similarity transformation, and finally a full affine warp. Here, tracking with one appearance model is used to initialize tracking with another. Based on this analogy, we add another step to the algorithm above:

Step 0: Track using the template $T_n(\mathbf{x}) = I_{n-1}(\mathbf{W}(\mathbf{x};\mathbf{p}_{n-1}^*))$ with *no* appearance model.

This step is performed before the two steps above and is used to initialize them.

3.2 Quantitative Evaluation

We evaluate Strategy 4 "Template and Appearance Model Update with Drift Correction" in the same way that we evaluated Strategy 3 in Section 2.4. We use a 947 frame video of a face and construct an initial AAM for it by hand-marking feature points in a random selection of 80 frames. We then generate ground-truth parameters by tracking the AAM through the video using a combination of AAM fitting [8], pyramid search, progressive transformation complexity, and re-initialization by hand. The accompanying movie "`face-gt.mpg`" plots the ground-truth AAM feature points on all images in the video sequence. The sequence shows a drivers face in a car and includes moderate face pose and lighting variation. We generate 50 test cases for each of the 947 frames in the video by randomly perturbing the AAM parameters. Similarly to Section 2.4, and following the exact procedure in [8], we generate perturbations in both the similarity transform of the AAM and the shape parameters. Specifically, the RMS similarity displacement standard deviation is chosen to be 4 times the shape eigenvector standard deviation so that each is weighted according to their relative importance. For each test case, we compared four algorithms:

Algorithm 1: Step 2 (no update).

Algorithm 2: Step 1 followed by Step 2.

Algorithm 3: Step 0 followed by Step 1 followed by Step 2.

Algorithm 4: Step 0 followed by Step 2.

We plot the frequency of convergence of these four algorithms computed on average across all 50×947 test cases against the magnitude of the perturbation to the AAM parameters in Figure 4. As for the single template tracking case in Section 2, the template

and appearance model update algorithms (Algorithms 2, 3, and 4) all outperform the algorithm which does not update the template and appearance mode (Algorithm 1). As one might imagine, Algorithm 3 (Steps 0, 1, 2) marginally outperforms Algorithm 2 which just uses Steps 1 and 2. Algorithm 4 performs significantly worse than both Algorithms 2 and 3 indicating that Step 1 is essential for the best performance.

3.3 Converting a Generic AAM to a Person-Specific AAM

When we use Step 1 above, a new template and appearance model are computed online as we track the face through the video. To illustrate this process we applied Algorithm 2 to track a video of a face using a generic, person-independent AAM. The accompanying movie "`face-app.mpg`" shows the tracked face, $T_1(\mathbf{x})$ and the first two $A_1^i(\mathbf{x})$. Also shown are the current $T_n(\mathbf{x})$ and the first two $A_n^i(\mathbf{x})$ for each frame. The result is that at the end of the sequence, the template and appearance model update algorithm has computed a person specific appearance model.

This process is illustrated in Figure 5. Figure 5(a) shows 4 frames of the face that is tracked. Note that no images of the person in the video were used to compute the generic AAM. Figure 5(b) shows the appearance eigenvectors of the generic AAM. Note that the appearance eigenvectors contain both identity and illumination variation. Figure 5(c) shows the appearance eigenvectors of the person-specific AAM computed using our algorithm. Note that the eigenvectors mainly code illumination variation, and no identity variation. Figure 5(d) plots the appearance eigenvalues of both AAMs. There is far less appearance variation in the person-specific AAM and it therefore requires far fewer appearance parameters to provide the same representational power.

4 Summary

We have investigated the template update problem. We first proposed a template update algorithm that does not suffer from the "drift" inherent in the naive algorithm. Next, we showed how this algorithm can be re-interpreted as a heuristic to avoid local minima and quantitatively evaluated it as such. The results show that updating the template using "Template Update with Drift Correction" improves tracking robustness. We then extended our algorithm to template tracking with linear appearance models and quantitatively compared four variants of the update strategy. The results again show that updating both the template and the appearance model with drift correction results in more robust fitting. Finally, we showed that our linear appearance model update strategy can also automatically compute a person-specific AAM while tracking with a generic AAM.

Acknowledgments

The research described in this report was partially supported by Denso Corporation, Japan, and was conducted at Carnegie Mellon University Robotics Institute while Takahiro Ishikawa was a Visiting Industrial Scholar from Denso Corporation. This research was also supported, in part, by the U.S. Department of Defense through award number N41756-03-C4024. The Generic AAM model in Section 3.3 was trained on the ViaVoice^{TM}AV database provided by IBM Research.

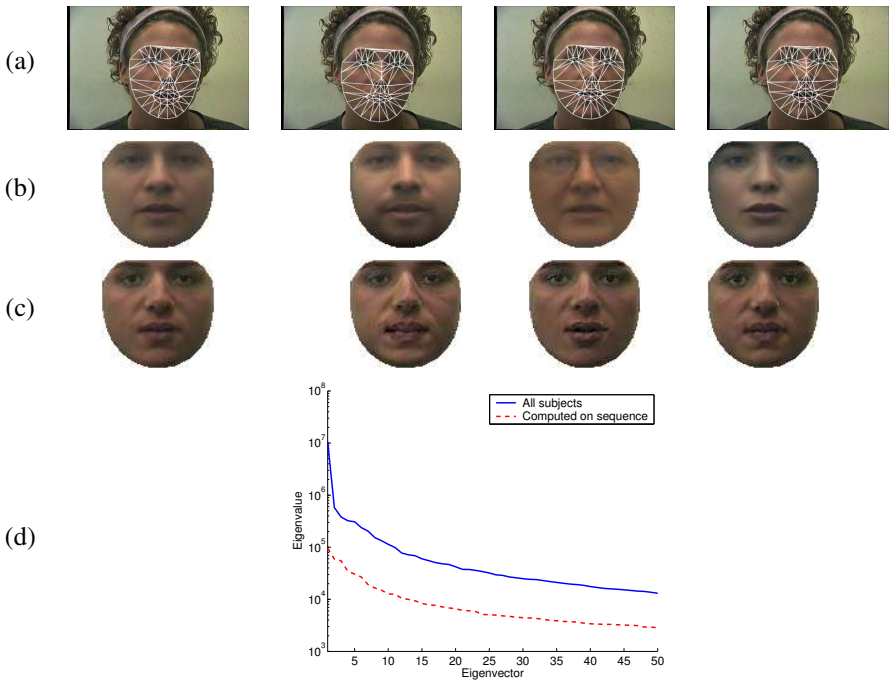


Figure 5: An illustration of the conversion of a generic AAM to a person-specific AAM. (a) Four frames from the video that is tracked. (b) The appearance variation of the generic AAM. (c) The appearance variation of the person-specific AAM. (d) The appearance eigenvalues of the two AAMs.

References

[1] S. Baker and I. Matthews. Equivalence and efficiency of image alignment algorithms. In *Proc. of CVPR*, volume 1, pages 1090–1097, 2001.

[2] S. Baker and I. Matthews. Lucas-Kanade 20 years on: A unifying framework: Part 1: The quantity approximated, the warp update rule, and the gradient descent approximation. *International Journal of Computer Vision*, 2003.

[3] J.R. Bergen, P. Anandan, K.J. Hanna, and R. Hingorani. Hierarchical model-based motion estimation. In *Proc. of ECCV*, pages 237–252, 1992.

[4] M. Black and A. Jepson. Eigen-tracking: Robust matching and tracking of articulated objects using a view-based representation. *IJCV*, 36(2):63–84, 1998.

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

[6] G. Hager and P. Belhumeur. Efficient region tracking with parametric models of geometry and illumination. *IEEE Trans Pattern Analysis and Machine Intelligence*, 20(10):1025–1039, 1998.

[7] B. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proc. of IJCAI*, pages 674–679, 1981.

[8] I. Matthews and S. Baker. Active appearance models revisited. Technical Report CMU-RI-TR-03-02, Carnegie Mellon University, Robotics Institute, 2003.

[9] S. Sclaroff and J. Isidoro. Active blobs. In *Proc. of the 6th IEEE International Conference on Computer Vision*, pages 1146–1153, 1998.