Tracking Using Dynamic Programming for Appearance-Based Sign Language Recognition

Philippe Dreuw, Thomas Deselaers, David Rybach, Daniel Keysers, and Hermann Ney
Lehrstuhl für Informatik VI – Computer Science Department
RWTH Aachen University – D-52056 Aachen, Germany
{dreuw,deselaers,rybach,keysers,ney}@cs.rwth-aachen.de

Abstract

We present a novel tracking algorithm that uses dynamic programming to determine the path of target objects and that is able to track an arbitrary number of different objects. The traceback method used to track the targets avoids taking possibly wrong local decisions and thus reconstructs the best tracking paths using the whole observation sequence. The tracking method can be compared to the nonlinear time alignment in automatic speech recognition (ASR) and it can analogously be integrated into a hidden Markov model based recognition process. We show how the method can be applied to the tracking of hands and the face for automatic sign language recognition.

1 Introduction

In automatic sign language recognition usually special devices such as data gloves, colored gloves, or wearable cameras are used to recognize gestures, and the gestures are often performed in front of a blue screen under normalized conditions. In a real environment, however, we are confronted with inhomogeneous background, occlusions, and further problems that are neither expected nor can be modelled. Under realistic circumstances, the performance of most current approaches decreases dramatically as it heavily depends upon possibly wrong local decisions.

To tackle these problems we avoid preliminary decisions and propose to use the same techniques that are successfully applied in automatic speech recognition (ASR). That is, we propose to use dynamic programming to implement a novel tracking algorithm. It is able to track an arbitrary number of different objects at the same time with basically no computational extra cost when the objects to be tracked are of the same type.

Related Work. When detailed information about moving objects in video sequences is needed, tracking comes into play. Popular tracking methods are the Condensation tracking [6], Kalman filtering [9], Meanshift tracking [5], and Camshift tracking [3]. In [10] face, torso, legs, or hands are detected and tracked in cluttered scenes using Boosting. In [2] a linguistic feature vector is used to recognize sign language. In [13] an algorithm for finding and kinematically tracking multiple people in long sequences is presented. Most of these approaches have in common that they make possibly wrong local decisions. A similar approach is presented in [1] where a dynamic programming framework is used to localize and recognize dynamic hand gestures, but we present a more general framework with the possibility to integrate multiple scoring functions e.g. Eigenfaces, or arbitrary objects, and the possibility to track multiple objects at the same time.

2 Tracking Using Dynamic Programming

The proposed tracking algorithm prevents taking possibly wrong local decisions, because the tracking is done at the end of a sequence by tracing back the decisions to reconstruct the best path. The best path is the path with the highest score wrt. a given scoring function. This procedure is related to time alignment in speech recognition. The tracking method can be seen as a two step procedure: in the first step, a score function is calculated for each frame starting from the first, and in the second step, the globally optimal path is traced back from the last frame of the sequence to the first.

Step 1. For each pixel (x, y) of a frame X_t at time step $t = 1, ..., T$ a score $q(t, x, y)$ is calculated, called the local score. Local score functions take into account the image X_t at time step t and the position (x, y). A global score $Q(t, x, y)$ is calculated for each time step t and each position (x, y). $Q(t, x, y)$ is the total score for the best tracking

until time step t which ends in position (x, y). Additionally, for each position (x, y) in image X_t, the best predecessor is searched among a set of possible predecessors from the scores $Q(t-1, x', y')$ within a neighborhood of position (x, y). This best predecessor is then stored for each time step t and each pixel (x, y) in a table of backpointers $B(t, x, y)$ which is used for the traceback in step 2. The recursive equation for this dynamic programming tracking algorithm is defined as follows:

$$\begin{aligned} Q(t,x,y) &= \max_{x',y' \in M(x,y)} \{(Q(t-1,x',y') \\ &\quad -\mathcal{T}(x',y',x,y)\} + q(t,x,y) \qquad (1) \\ B(t,x,y) &= \operatorname*{argmax}_{x',y' \in M(x,y)} \{(Q(t-1,x',y') \\ &\quad -\mathcal{T}(x',y',x,y)\} \qquad (2) \end{aligned}$$

where $M(x, y)$ is the set of possible predecessors of point (x, y) and $\mathcal{T}(x', y', x, y)$ is the jump-penalty from point (x', y') in the predecessor image to point (x, y) in the current image. As jump-penalty functions $\mathcal{T}$, for example the Euclidean distance $\mathcal{T}(x', y', x, y) = \alpha \cdot \sqrt{(x-x')^2 + (y-y')^2}$ or the L_1-norm can be used. α is a weighting factor to be able to control the impact of the penalty function with respect to the impact of the local score function $q(t, x, y)$. These simple penalty functions can be replaced by more complex motion models, e.g. taking into account hand or head positions.

Step 2. The traceback process reconstructs the best path $t \to (x, y)$ using the score table Q and the backpointer table B. A full traceback starts from last frame of the sequence at time step T using $c_T = \operatorname{argmax}_{x,y} Q(T, x, y)$ as tracking center. From the backpointer table, the best tracking center at time step $t-1$ is then obtained by $c_{t-1} = B(t, c_t)$. This process is iterated up to time step $t = 1$ to reconstruct the best path.

Using a full traceback, the tracking decision at time step t' thus depends on all scores of images from $t = 1, \ldots, T$. To track multiple identical objects one simply has to traceback multiple hypotheses where special care has to be taken when the paths (hypotheses) are overlapping.

A partial traceback at time step t' uses only the information from $t = 1$ up to $t = t' + \Delta$ with a reasonably chosen Δ, and can be calculated before the entire sequence is known, i.e. after only a short delay Δ (which could also be chosen to be $\Delta = 0$). One might argue that sign language recognition requires online tracking, but also in ASR a partial traceback of the decisions occurs to use the context of the observations.

Tracking using dynamic programming performs well when one wants to track an object in the presence of many occlusions, gaps, or for off-line tracking. It can also be used with non-static background or multiple target objects in the foreground. The tracking can then be controlled by the tracking rectangle sizes I and J, the predecessor area M, and the jump-penalty function $\mathcal{T}$. Dynamic programming tracking on large images can be very time consuming when performing a full search over all possible tracking rectangles, depending on the score function and the predecessor area. Thus, first we develop a tracking algorithm with fixed tracking rectangle size to reduce the number of hypotheses that we have to consider. In Figure 1 the complete tracking is illustrated. It shows a tracked image sequence consisting of 68 frames at different time steps $t = 10, 14, 18, 22, 26$. In the first step, the score function is propagated from left to right and in the second step, the traceback is done from right to left.

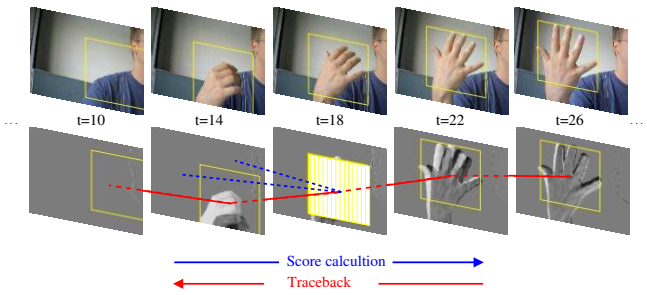

Figure 1. Tracking of a hand signing the gesture for the German letter combination "SCH". The dynamic programming algorithm uses skin color difference images to calculate the scores for the traceback.

Fine Tuning. As not each possible tracking center is likely to produce a high score, pruning can be integrated into the dynamic programming tracking algorithm for speed-up. At time step $t = 0$ each point (x, y) is initialized using the local score $q(t = 0, x, y)$ and all points are activated as possible predecessors for points at time step $t = 1$. From this time step onwards until the end of the sequence, a point (x, y) will only be considered as a predecessor for time step $t+1$ if $Q(t, x, y) > \max_{x,y}(Q(t, x, y)) - T_0$ holds for a suitable pruning threshold T_0.

In many cases it might be necessary to allow a change of the tracking box size, which means a full search over all possible predecessors for the upper left and bottom right corner of the tracking rectangle. In this case we only have to adapt our recursive equation and jump penalty function. As a full search in this setting will run into serious runtime problems, we use locally adaptive tracking rectangles (at each time step we optimize the size only locally, keeping the global decision for the sequence of tracking centers).

3 Score Functions

In this section we present score functions starting from a simple one which uses only difference images and finally we present a score function that uses eigenfaces and skin color information to track faces and hands in video sequences of sign language utterances.

Motion Information Score Function. One possible way to track objects is to assume that the objects to be tracked are moving and to look at difference images where motion occurs and to track these positions. That is, using a first-order time derivative $\widetilde{X_t}$ of an image X_t as image feature, the local score can be calculated by a weighted sum over the absolute pixel values inside the tracking area:

$$q(t,x,y) = \sum_{\substack{x'=x-(I-1)/2 \\ y'=y-(J-1)/2}}^{\substack{x'=x+(I-1)/2 \\ y'=y+(J-1)/2}} w_{y'} \cdot w_{x'} \cdot |\widetilde{X_t}(x',y')|, \quad (3)$$

$$w_{y'} = 1.5 - \frac{|y'-y|}{J/2}, w_{x'} = 1.5 - \frac{|x'-x|}{I/2}.$$

In this simple case, the minimum search window size $I \times J$ must be at least 3×3 in order to center the window.

The score function can be replaced by any other score function. For example, the distance between a target model and a target candidate could be defined here. In the following we present how to incorporate face detection by eigenfaces into this method.

Eigenfaces and Skin Color Score Functions. Turk and Pentland applied principal component analysis to face recognition and detection [15]. Principal component analysis is performed on a set of training images showing faces to generate their eigenvectors (here called eigenfaces) which span a subspace (called the face space). To detect whether an image patch shows a face or not, it is projected into the subspace and back-projected using e.g. only the 20 first face space components. Then, the distance between the original image and the back-projection can be calculated. This can be efficiently done for complete images using the fast Fourier transformation and the remaining energy in the subspace. Due to the nature of the eigenfaces, face-like images can be reconstructed well, whereas non-faces are reconstructed poorly and thus the distance between the original image and the back-projection is high in this case, or the energy in the subspace is low. This distance can be seen as a measure of *faceness* and can thus be used as a local score function $q_f(t,x,y)$ for tracking faces. That is, the faceness information can directly be incorporated into the above described tracking process.

To make this method more stable, we further want to use color information from the images to help the method in recognizing faceness. As faces generally are skin colored, we use a skin color model to determine whether a position (x,y) in an image is skin colored or not [8].

Figure 2. Tracking using dynamic programming with eigenfaces and skin color probability scoring functions: the first image shows that using only eigenfaces to detect or recognize faces in Sign Language is insufficient due to occlusions of hand or inclined head position, the second that combining the eigenface scoring function with skin color information strongly improves the result. The third shows an example of multiple tracking of head and hands using eigenfaces, skin color probability and motion scoring functions.

To combine these two methods, we redefine our local score function to $q(t,x,y) = (1-w) \cdot q_s(t,x,y) + w \cdot q_f(t,x,y)$ where $q_s(t,x,y)$ is the score function obtained from the skin color model and $q_f(t,x,y)$ is the faceness score function. w is a weighting factor. Some scoring function examples are shown in Figure 2.

To train the eigenfaces, we used the BioID database[1]. Using the combined score function and pruning, we improved the runtime as usually only regions with a high skin color probability contain faces.

4 Comparison of Dynamic Programming Tracking to Other Approaches

In this section we compare our proposed tracking algorithm to time alignment in speech recognition and to the Condensation tracking algorithm. The comparison to the time alignment is especially interesting as the tracking will later be integrated into the hidden Markov based decision process as the time alignment is in ASR.

Comparison to Time-Alignment. In automatic speech recognition, it is unclear at which time t in a sequence of observation vectors $X_1, \ldots X_T$ a word starts and where it ends. So correspondences among observation sequences have to be determined. This process is shortly outlined in the following:

Given two sequences of vectors over the time axes t and s, $X_1^T = (X_1 \ldots X_t \ldots X_T)$, $X_t \in \mathbb{R}^D$ and $Y_1^S = (Y_1 \ldots Y_s \ldots Y_S)$, $Y_s \in \mathbb{R}^D$, we want to find an optimal mapping $t \mapsto s(t)$ of "corresponding" vectors. This

[1] http://www.bioid.com

task is referred to as time alignment in automatic speech recognition, in which case X_1^T is the observation sequence and Y_1^S is one possible reference sequence.

The standard technique used in automatic speech recognition is the nonlinear time alignment. The set of allowed paths s_1^T is constrained by the model topology. For the $(0, 1, 2)$-standard model the optimization criterion is:

$$\min_{t \to s(t)} \sum_{t=1}^{T} \left[d(X_t, Y_{s(t)}) + \mathcal{T}(s(t) - s(t-1)) \right]$$

with the time distortion penalty $\mathcal{T}(s(t) - s(t-1))$. (We can interpret the model as having infinite distortion penalties $\mathcal{T}(\delta) = \infty$ for $\delta \notin \{0, 1, 2\}$.) The model topology here is related to the predecessor area M and the time distortion penalty is related to the jump-penalty function $\mathcal{T}$ defined in equation (1).

Using the (0,1,2)-standard model and dynamic programming to solve the optimization criterion, we obtain the following recursive equations:

$$\begin{aligned} D(t,s) &= \min_{\delta \in \{0,1,2\}} \{D(t-1, s-\delta) + \mathcal{T}(\delta)\} \\ &\quad + d(X_t, Y_s) \qquad (4) \\ B(t,s) &= \operatorname*{argmin}_{\delta \in \{0,1,2\}} \{D(t-1, s-\delta) + \mathcal{T}(\delta)\} \qquad (5) \end{aligned}$$

We can see that the recursive equations (4) and (5) correspond to the equations (1) and (2), respectively. This relation opens up the possibility to directly integrate the tracking into the recognition process in the same way the time alignment is integrated in the ASR process.

Comparison to Condensation Tracking. The Condensation tracking algorithm presented in [6] uses a probabilistic framework for tracking in the presence of clutter using random sampling. The object dynamics form a temporal Markov chain and the observation probabilities are propagated by the Bayes' rule.

The Condensation algorithm models the object dynamics. By sampling from the learned shape and motion model a new sample-set is predicted. The number of used samples in each time step to track an object has a high impact on the accuracy and the runtime of the tracking algorithm. This method allows for recovering from a temporary tracking failure.

In contrast to this, using dynamic programming for tracking (DPT), no motion model is estimated but jump-penalty functions which can be extended and adapted to approximate the usual motion of a target object are used. If pruning is used, and the best tracking path is lost, a recovery is not possible anymore, if no pruning is used, the optimal path is guaranteed to be found.

If we included an additional state variable in the dynamic programming algorithm that encoded the object dynamics, we would be able to recover the overall best path taking the object dynamics into account. As a complete search over such a large state space is infeasible, the condensation algorithm uses pruning to reduce the number of active states (samples) in each step. To allow for recovering possibly lost sequences, random restarts are allowed. In contrast to this, our dynamic programming algorithm prunes the large state space by unifying all states that have different object dynamics but instead allows a larger number of active hypotheses in each step and does not introduce the element of randomness used in the Condensation algorithm. Thus, the dynamic programming tracking algorithm allows us to track an object even if it is occluded for a prolate time or if it shows motion unexpected by the motion model, which is a clear advantage over the condensation tracking algorithm. Furthermore, another advantage is that the DPT algorithm allows for tracking multiple objects with basically no additional costs. The only extra effort that has to be taken to track multiple objects is that a very inexpensive traceback is necessary per object. The more expensive calculation of the scores $Q(t, x, y)$ is necessary only once.

5 Experimental Results

The advantage of considering the whole sequence before making any decision becomes apparent when tracking has to be done under noisy circumstances.

Figure 3 shows the dynamic programming tracking compared to the Camshift tracker on a sequence with different noise levels. Both trackers are tracking a hand gesturing a "Z" of the German finger-spelling alphabet using first-order time derivative images of the original images thresholded by skin color probability. The plot in each row represents a smoothed trajectory of the tracking rectangle centers and should contain an S-shaped trajectory, i.e. a mirror-inverted Z-shape. Our proposed tracking algorithm still is able to reasonably track the hand at a very high noise-level whereas the Camshift tracker is no longer able to follow the hand due to reduced skin color information because of the high noise level in the image sequences. The sequences for these tests were taken from a database of videos showing the German fingerspelling alphabet[2]. It can clearly be observed, that the dynamic programming tracking is more robust to noise than the Camshift tracking.

6 Combining Recognition and Tracking

Hidden Markov models (HMMs) are the standard method to compensate for time and amplitude variations.

[2] http://www-i6.informatik.rwth-aachen.de/~dreuw/database.html

Figure 3. Comparing DPT and Camshift tracking. All example images are showing a person at time step $t = 4, 11, 18, 28, 36$ signing the letter "Z" in German Sign Language and are disturbed by different noise-levels. In each line, f.l.t.r.: tracking result images of DPT, a smoothed trajectory plots of the DPT tracking rectangle centers, and in reverse order the result of tracking the same sequence using Camshift tracking. Lines (a),(b),(b) show the results of tracking the same sequence under different noise levels.

HMMs are generally used for ASR [7], gesture recognition [12], sign language recognition [14, 16], and human action recognition [4, 11].

In ASR, recognition and time alignment are an interwoven process. The entangledness of time alignment and recognition allows for example to prune unlikely hypotheses and thus fewer hypotheses have to be considered. Through the close relation between dynamic programming tracking and time alignment it is possible to benefit from the same effects in gesture recognition. In this section we briefly explain how tracking and recognition of gestures can be directly combined thus benefiting from the experiences in automatic speech recognition.

To classify an observation sequence X_1^T, Bayes' decision rule is used:

$$\begin{aligned} X_1^T \longrightarrow r(X_1^T) &= \underset{k}{\operatorname{argmax}} \left\{ p(k|X_1^T) \right\} \\ &= \underset{k}{\operatorname{argmax}} \left\{ p(k) \cdot p(X_1^T|k) \right\} \\ &= \underset{k}{\operatorname{argmax}} \left\{ p(k) \cdot \sum_{s_1^T} \left\{ \prod_{t=1}^{T} p(X_t, s_t | X_1^{t-1}, s_1^{t-1}, k) \right\} \right\} \end{aligned}$$

where X_1^T is a sequence with images $X_1, \ldots, X_T$. Here, $p(k)$ is the a priori probability of class k, $p(X_1^T|k)$ is the class conditional probability for the observation X_1^T given class k and $r(X_1^T)$ is the decision of the classifier.

To integrate the dynamic programming tracking into the recognition process, we rewrite the class specific densities as follows:

$$\begin{aligned} r(X_1^T) &= \underset{k}{\operatorname{argmax}} \left\{ p(k) \cdot p(X_1^T|k) \right\} \\ p(X_1^T|k) &= \sum_{[s_1^T, l_1^T]} p(X_1^T, s_1^T, l_1^T|k) \\ p(X_1^T, s_1^T, l_1^T|k) &= \prod_{t=1}^{T} p(X_t, s_t, l_t | X_1^{t-1}, s_1^{t-1}, l_1^{t-1}, k) \quad (6) \end{aligned}$$

with l_1^T a sequence over time of the tracking states $l_1, ..., l_t, ..., l_T$.

Here, a tracking state l_t can be a simple location (x_t, y_t), e.g. the center of a tracking rectangle, a location with a specific range (x_t, y_t, r_t), e.g. the center of a tracking rectangle of size r_t, or any more complex description of the current tracking situation $(x_t, y_t, r_t, ...)$ including e.g. the dynamics.

Combining these two processes can be very time consuming and additional assumptions have to be made. We assume that the probability $p(X_t, s_t, l_t | X_1^{t-1}, s_1^{t-1}, l_1^{t-1}, k)$ only depends on the abstract states $s = s_1, ..., s_T$ of the gesture classes k (which means "hidden" states). Furthermore, it is assumed that the transition probabilities depend only on the predecessor state, and that the emission probabilities depend on the reached state: We can simplify equation (6) now as follows:

$$\begin{aligned} & p(X_t, s_t, l_t | X_1^{t-1}, s_1^{t-1}, l_1^{t-1}, k) \\ = \; & p(X_t, s_t, l_t | s_1^{t-1}, l_1^{t-1}, k) \\ = \; & p(X_t, s_t, l_t | s_{t-1}, l_{t-1}, k) \\ = \; & \underbrace{p(s_t, l_t | s_{t-1}, l_{t-1}, k)}_{\text{Transition probability}} \cdot \underbrace{p(X_t | s_t, l_t, k)}_{\text{Emission probability}} \end{aligned}$$

Additionally, we assume that the transition probability is independent of the tracking state, i.e.

$$p(s_t, l_t | s_{t-1}, l_{t-1}, k) = p(s_t | s_{t-1}, k) \cdot p(l_t | s_{t-1}, l_{t-1}, k)$$

and that the tracking state probability only depends on the predecessor tracking state, i.e.

$$p(s_t, l_t | s_{t-1}, l_{t-1}, k) = p(s_t | s_{t-1}, k) \cdot p(l_t | l_{t-1}, k)$$

Combining DPT and recognition into an HMM changes the modeling of the emission probability. We can model the tracking state probability as $-\log(p(l_t|l_{t-1}, k)) = \alpha \cdot$

$||(l_t - l_{t-1}) - \mu^l_{s_{t-1}}||^2$ which considers the Euclidean distance between the current tracking state and the predecessor and additionally estimates a tracking state model $\mu^l_{s_{t-1}}$ in the HMM.

This proposed method allows us to have tracking as an integral part of the recognition process and thus to avoid possibly wrong local decisions.

7 Summary and Conclusions

We proposed a new dynamic programming tracking algorithm based on the idea of time alignment in speech recognition. We compared the new algorithm to the Condensation algorithm theoretically and showed its practical potential under noisy circumstances in comparison to the Camshift tracking algorithm. In this comparison, the proposed algorithm showed superior performance.

The comparison to condensation tracking or camshift might seem unfair but shows the advantage of using more context information. These approaches are online tracking and do not know the whole sequence while the DPT allows also partial tracebacks over e.g. only one or two frames.

Our DPT framework can be extended by incorporating any image comparison function to calculate the distance between a target model and a target candidate, and different motion models can be easily integrated in our framework. Using the information of the entire sequence by tracing back the decisions at the end of the sequence reconstructs the best path and enables us to track a target disregarding information gaps in the video sequence due to occlusions or strong noise. Integrating eigenfaces in combination with skin color information into our tracking framework enables to track hands and faces for automatic sign language recognition.

Additionally we proposed a framework to integrate the dynamic programming tracking algorithm into an HMM to allow a simultaneous tracking and recognition in one process which is interesting for further research.

References

[1] J. Alon, V. Athitsos, Q. Yuan, and S. Sclaroff. Simultaneous Localization and Recognition of Dynamic Hand Gestures. In *IEEE WMVC*, vol. 2, pages 254–260, Jan 2005. 1

[2] R. Bowden, D. Windridge, T. Kadir, A. Zisserman, and M. Brady. A Linguistic Feature Vector for the Visual Interpretation of Sign Language. In J. M. Tomas Pajdla, editor, *ECCV*, vol. 1, Prague, Czech Republic, pages 391–401, May 2004. 1

[3] G. R. Bradski. Computer Vision Face Tracking For Use in a Perceptual User Interface. *Intel Technology Journal*, Q2:15–26, 1998. 1

[4] M. Brand, N. Oliver, and A. Pentland. Coupled hidden Markov models for complex action recognition. In *IEEE CVPR*, IEEE Computer Society, Washington, DC, USA, pages 994–, Jun 1997. 5

[5] D. Comaniciu, V. Ramesh, and P. Meer. Real-Time Tracking of Non-Rigid Objects using Mean Shift. In *IEEE CVPR*, vol. 2, Hilton Head Island, South Carolina, USA, pages 142–151, Jun 2000. 1

[6] M. Isard and A. Blake. CONDENSATION – conditional density propagation for visual tracking. *IJCV*, 29(1):5–28, Aug 1998. 1, 4

[7] F. Jelinek. *Statistical Methods for Speech Recognition*. MIT Press, Cambridge, Massachusetts, Jan 1998. 5

[8] M. Jones and J. Rehg. Statistical Color Models with Application to Skin Color Detection. Technical Report CRL 98/11, Compaq Cambridge Research Lab, 1998. 3

[9] R. E. Kalman. A New Approach to Linear Filtering and Prediction Problems. *Transaction of the ASME - Journal of Basic Engineering*, 82:35–45, Mar 1960. 1

[10] A. Micilotta, E. Ong, and R. Bowden. Detection and Tracking of Humans by Probabilistic Body Part Assembly. In *BMVC*, number 1, Oxford UK, pages 429–438, Sept 2005. 1

[11] N. Nguyen, H. Bui, S. Venkatesh, and G. West. Recognising and monitoring highlevel behaviours in complex spatial environments. In *IEEE CVPR*, vol. 2, Madison, Wisconsin, pages 620–625, Jun 2003. 5

[12] P. Dreuw, D. Keysers, T. Deselaers, and H. Ney. Gesture Recognition Using Image Comparison Methods. In *International Gesture Workshop 2005*, LNAI, vol. 3881, Île-de-Berder, France, pages 124–128, May 2005. 5

[13] D. Ramanan, D. A. Forsyth, and A. Zisserman. Strike a Pose: Tracking People by Finding Stylized Poses. In *Proceedings of the IEEE CVPR*, vol. 1, San Diego, USA, pages 271–278, 2005. 1

[14] T. Starner, J. Weaver, and A. Pentland. Real-time American sign-language recognition using desk and wearable computer based video. *IEEE PAMI*, 20(12):1371–1375, Dec 1998. 5

[15] M. Turk and A. Pentland. Eigenfaces for Recognition. *Journal of Cognitive Neuroscience*, 3(1):71–86, Jan 1991. 3

[16] C. Vogler and D. Metaxas. A Framework for Recognizing the Simultaneous Aspects of American Sign Language. *CVIU*, 81(3):358–384, Mar 2001. 5