Proc. ACCV 2010, Springer LNCS 6494 pp. 422-435, DOI: 10.1007/978-3-642-19318-7_33

Affine Warp Propagation for Fast Simultaneous Modelling and Tracking of Articulated Objects

Arnaud Declercq and Justus Piater

Montefiore Institute, University of Liège, Belgium
Arnaud.Declercq@ulg.ac.be, Justus.Piater@ulg.ac.be

Abstract. We propose a new framework that allows simultaneous modelling and tracking of articulated objects in real time. We introduce a non-probabilistic graphical model and a new type of message that propagates explicit motion information for realignment of feature constellations across frames. These messages are weighted according to the rigidity of the relations between the source and destination features. We also present a method for learning these weights as well as the spatial relations between connected feature points, automatically identifying deformable and rigid object parts. Our method is extremely fast and allows simultaneous learning and tracking of nonrigid models containing hundreds of feature points with negligible computational overhead.

1 Introduction

Articulated object models have become an active research topic in recent years. In the vast majority of applications, an object is represented by a graphical model connecting rigid body parts [1, 2]. While those models are usually designed by hand, some solutions have been proposed to automatically learn articulated models from tracks of feature points [3, 4]. Unfortunately, robust feature tracking proves to be a challenge on its own in long sequences.

Tracking those points becomes much easier when they can rely on a feature graph to assist them. Since good tracking requires a model and a model is learnt based on good tracking, the most obvious solution is to simultaneously track and learn the feature graph. While some solutions have already been proposed for offline learning [5] or learning based on a short initialisation period [6], very few are dedicated to online learning. This domain is indeed very challenging: not only must both learning and tracking be computed in real time, but also tracking must rely on an incomplete intermediate model. Although we addressed the latter issue with an uncertain Gaussian model that explicitly accounts for its predictive power [7], we were then only able to achieve real-time tracking on very small feature graphs. Most of the computational time was not dedicated to learning but to tracking using the popular Belief Propagation solution [8, 9] to propagate position likelihoods between nodes of the graph.

Here, we propose a new tracking solution that sacrifices the multimodality of non-parametric, probabilistic methods for a significant gain in computational efficiency. The

main idea here is to keep the benefits of a propagation scheme to share information between nodes, but formulated for a new, non-probabilistic feature graph. While each node of the graph still represents the current position of its associated feature point, propagated messages do not convey a potential function but simply the information needed to align feature points in the new image. Thanks to this solution, we will show that large feature graphs can be simultaneously learnt and tracked in real time. As feature points, we will present our method using edgels (i.e. points along edges) because they are more robust to illumination changes and lack of texture. This also allows us to show that, even if each edgel only provides a 2D translational constraint, it can be tracked properly in a 6-dimensional affine space thanks to our propagation method.

After discussing some background in Section 2, we will introduce in Section 3 the image alignment of a template based on the motion of a complete set of feature points. This will give us a first idea of the information that has to be conveyed in order to align features in a new image. In Section 4, we will consider each feature as a template but with a connection only to its direct neighbours, and show how information from further features can be propagated in order to be used in its alignment. In Section 5, we explain how spatial relations between feature points are learnt and used during Affine Warp Propagation. Finally, experimental results are presented in Section 6.

2 Related Work

Various methods have been proposed for simultaneous learning and tracking of rigid graph models [7, 10, 11]. The main drawback of these methods is their limitation to rigid objects or so small that feature displacements can be assumed spatially coherent.

Unsupervised learning of articulated models from a video sequence usually relies on existing feature trajectories that are processed off-line [4, 3]. Ramanan et al. [5] proposed an off-line unsupervised method that simultaneously discovers, tracks and learns articulated models of animals from video. Unfortunately, their method is slow and requires the whole video to be treated as a block, making it impossible to adapt to an on-line process. Krahnstoever et al. [6] presented an automatic on-line acquisition and initialisation of articulated models. Their method extracts independently moving surfaces and tracks them using Expectation-Maximisation during a short initialisation period. An articulated model is deduced from these motions and is used for tracking in subsequent images. Since no model update is provided, it is strongly dependent on the key-frames selected for learning. Finally, Droin et al. [12] developed a method to incrementally segment the rigid parts of an object on-line. It maintains a set of possible models learnt from previous frames, and uses them for tracking. Although interesting, their technique suffers two main drawbacks: it requires a foreground extraction, and it doesn't learn or use any spatial relations during tracking.

In terms of feature-graph tracking, the most popular solution is Belief Propagation (BP) [8, 9] and its derivatives like Nonparametric Belief Propagation or Sequential Belief Propagation [13, 14, 9] that combine BP with particle filters. Although our method is completely different from BP since it is a non-probabilistic solution, its message passing process is strongly inspired from it. Apart from that, our method is more closely related to image alignment techniques such as the Lucas-Kanade algorithm [15, 16].

3 Image Alignment via Affine Warps

Consider, first, that we have a model of the object we want to track. This model consists of a weighted set of n edgels and their positions $X = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ in the model's referential. In order to align this model with a set of target positions $T = (\mathbf{t}_1, \ldots, \mathbf{t}_n)$ in a given image, we use an affine warp:

$$W(\mathbf{x}_i; \mathbf{p}) = \begin{bmatrix} 1+p_1 & p_3 & p_5 \\ p_2 & 1+p_4 & p_6 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} \tag{1}$$

where $\mathbf{p} = [p_1, p_2, p_3, p_4, p_5, p_6]$ corresponds to the warp's parameters, and $\mathbf{x}_i = [x_i, y_i]$. Alignment of the model with target positions is given by the warp that minimises the sum of squared residuals

$$\mathbf{R} = \sum_i w_i \|W(\mathbf{x}_i; \mathbf{p}) - \mathbf{t}_i\|^2 \tag{2}$$

with w_i representing the weight of point i.

In the case of tracking, we can assume that a current estimate of $\mathbf{p}$ is known and target positions $T = (\mathbf{t}_1, \ldots, \mathbf{t}_n)$ can be obtained using the new image (using the closest contour points to each edgel for example). Alignment of the template in the new image is then obtained by minimising $\mathbf{R}$ for the incremental warp $W(\mathbf{x}, \Delta\mathbf{p})$:

$$\mathbf{R} = \sum_i w_i \|W(W(\mathbf{x}_i; \mathbf{p}); \Delta\mathbf{p}) - \mathbf{t}_i\|^2 \tag{3}$$

Then parameters $\mathbf{p}$ are updated such that

$$W(\mathbf{x}; \mathbf{p}_{\text{new}}) = W(W(\mathbf{x}; \mathbf{p}_{\text{old}}); \Delta\mathbf{p}). \tag{4}$$

In order to solve the warp update, the expression in equation 3 is linearised by performing a first-order Taylor expansion on $W(W(\mathbf{x}; \mathbf{p}); \Delta\mathbf{p})$ to give:

$$\mathbf{R} = \sum_i w_i \left\| W(W(\mathbf{x}_i; \mathbf{p}); \mathbf{0}) + \frac{\partial W}{\partial \mathbf{p}} \Delta\mathbf{p}_i - \mathbf{t}_i \right\|^2 \tag{5}$$

Since $W(\mathbf{x}; \mathbf{0})$ is the identity warp, $W(W(\mathbf{x}; \mathbf{p}); \mathbf{0})$ then simplifies to $W(\mathbf{x}; \mathbf{p})$. Following the notational convention that partial derivatives with respect to a column vector are laid out as a row vector and with $W(\mathbf{x}; \mathbf{p}) = [W_x, W_y]^T$, the Jacobian $\frac{\partial W}{\partial \mathbf{p}}$ of the warp at equation 5 is given by:

$$\frac{\partial W}{\partial \mathbf{p}} = \begin{bmatrix} \frac{\partial W_x}{\partial p_1} & \frac{\partial W_x}{\partial p_2} & \cdots & \frac{\partial W_x}{\partial p_6} \\ \frac{\partial W_y}{\partial p_1} & \frac{\partial W_y}{\partial p_2} & \cdots & \frac{\partial W_y}{\partial p_6} \end{bmatrix} = \begin{bmatrix} x & 0 & y & 0 & 1 & 0 \\ 0 & x & 0 & y & 0 & 1 \end{bmatrix} \tag{6}$$

Notice that $\frac{\partial W}{\partial \mathbf{p}}$ is computed for $W(W(\mathbf{x}; \mathbf{p}); \Delta\mathbf{p})$, which means that values used in equation 6 are the current coordinates of the points in the image.

The solution to the minimisation of equation 5 is obtained by setting to zero its partial derivatives with respect to $\Delta\mathbf{p}$:

$$\Delta\mathbf{p} = \mathbf{H}^{-1} \sum_i w_i \left[\frac{\partial W}{\partial \mathbf{p}}\right]^T D_i \tag{7}$$

where $D_i = [\mathbf{t}_i - W(\mathbf{x}_i; \mathbf{p})]^T$ is the displacement required of point i, and $\mathbf{H}$ is the $n \times n$ Hessian matrix (here with $n = 6$):

$$\mathbf{H} = \sum_i w_i \left[\frac{\partial W}{\partial \mathbf{p}}\right]^T \left[\frac{\partial W}{\partial \mathbf{p}}\right] \tag{8}$$

The warp update then consists of iteratively applying equations 7 and 4 until estimates of the parameters $\mathbf{p}$ converge (in the case of affine transformation, one iteration is sufficient since the system is linear in the parameters which wouldn't be the case with a parameter such as orientation for example).

The solution obtained in equations 7 and 8 is interesting because it means that $\Delta\mathbf{p}$ is computed using only the two matrices $\mathbf{H}$ and $\mathbf{S} = \sum_i w_i \left[\frac{\partial W}{\partial \mathbf{p}}\right]^T D_i$ whose sizes are independent of the number of points used. The fact that both matrices are computed as a sum over the points is also advantageous since information provided by new points will be easily added to the current matrices. Those two conditions met, a message containing them seems an attractive candidate to propagate motion information.

4 Affine Warp Propagation

In the case of a feature graph, no feature point is connected to a global model. Each one has only access to a set of learnt relations with its direct neighbours. Since the motion of an edgel is locally ambiguous, it will require information from the biggest possible neighbourhood. Moreover, if relations between edgels have to be learnt, it would be more convenient to express edgel configurations in an affine space instead of simply 2D space. In order to compute the affine warp needed to align a point and its surrounding in a new image, information will then have to be propagated in the graph. Before getting into the details of which information is required and how to propagate it, let's first consider the representation of the non-probabilistic graphical model we wish to learn.

4.1 Non-Probabilistic Graphical Model Definition

A node i of a graph contains the set of parameters $\mathbf{p}_i$ used to warp the corresponding edgel i and its neighbourhood from the origin to a position that aligns them with the current image. In that sense, a node is similar to the template discussed in Section 3 except that it doesn't have direct connection to all the points used for its alignment.

An edge going from node i to node j represents two types of information. First, it contains the learnt affine parameters $\mathbf{r}_{i|j}$ of i as they were previously observed in the affine space of j, i.e. the parameters that align i and its surroundings from the origin to

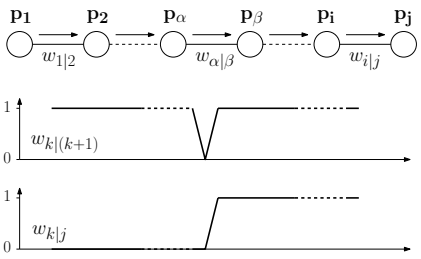

Fig. 1. Simple line graph situation where all spatial relations between connected points are rigid except for the relation between α and β. This means that this graph represents two uncorrelated rigid sets of points: one on the left and one on the right of the relation between α and β. The learnt weights between a node k and the node to its right are given by $w_{k|(k+1)}$ (centre panel). In the case where messages are propagated from left to right, the bottom panel illustrates the resulting influence $w_{k|j}$ on the rightmost point j of each upstream point k. Here, the null weight $w_{\alpha|\beta}$ eliminates the influence on p_j of the points that are not in the same rigid set.

their observed positions in the space of j. Secondly, it contains the weight $w_{i|j}$ node j should give to the information coming from node i. This weight is directly related to the rigidity of the relation: the lower the correlation between two features, the lower the weight and then the smaller the influence of messages passing through this connection. This means that information coming from points behind a non-rigid connection will have no influence on image alignment (see Figure 1 for an example).

4.2 Message Definition

In this section, we will use, for the sake of explanation, the simple line graph shown in Figure 1. If node j had access to the displacement D_k of all the points on the graph, its warp update using equation 7 would be

$$\Delta\mathbf{p}_j = \mathbf{H}_j^{-1} \sum_{k \leq j} w_{k|j} \left[\frac{\partial W_{k|j}}{\partial \mathbf{p}_j} \right]^T D_k, \tag{9}$$

where we use $\frac{\partial W_{k|j}}{\partial \mathbf{p}_j}$ to indicate that the Jacobian is computed for $W(W(\check{\mathbf{r}}_{k|j}; \mathbf{p}_j); \mathbf{0})$, i.e. for the projection in the image coordinates of the 2D position $\check{\mathbf{r}}_{k|j} = [r_{x,k|j}, r_{y,k|j}]$ of node k in the referential of node j. Now, if we define $w_{k|j} = w_{k|i} w_{i|j}$ for the graph of Figure 1, the sum can be decomposed the following way:

$$\mathbf{S}_j = \sum_{k \leq j} w_{k|j} \left[\frac{\partial W_{k|j}}{\partial \mathbf{p}_j} \right]^T D_k \tag{10}$$

$$= w_j \left[\frac{\partial W_{j|j}}{\partial \mathbf{p}_j} \right]^T D_j + w_{i|j} \sum_{k \leq i} w_{k|i} \left[\frac{\partial W_{k|i}}{\partial \mathbf{p}_i} \right]^T D_k \tag{11}$$

$$= S_j + w_{i|j} \mathbf{S}_i \tag{12}$$

where $w_j = w_{j|j}$ is the weight node j gives to its feature point and, more importantly, where we made the assumption that

$$\frac{\partial W_{k|j}}{\partial \mathbf{p}_j} = \frac{\partial W_{k|i}}{\partial \mathbf{p}_i} \tag{13}$$

This assumption means we consider that point k is projected in the same image coordinates by nodes i and j, i.e. $W(\check{\mathbf{r}}_{k|j}; \mathbf{p}_j) = W(\check{\mathbf{r}}_{k|i}; \mathbf{p}_j)$ which is actually correct if nodes i and j are linked by a perfectly rigid spatial relation. Since the weights are null for non-rigid relations, this assumption seems valid. Unfortunately, we will see in Section 4.3 that even if the weights are all correct (which is not guaranteed during the learning phase), small numerical errors can trigger a drift from the correct tracking result. While this will motivate a more general formulation in Section 4.3, we propose to continue with this assumption for now in order to understand more easily what type of information a message should contain.

Decomposing $\mathbf{H}_j$ in a similar way to Equation 12, we obtain:

$$\mathbf{H}_j = H_j + w_{i|j}\mathbf{H}_i \tag{14}$$

Equations 12 and 14 mean that the warp update for a node j based on all the points of the graph in Figure 1 can be computed using only the information from itself and that accumulated by node i. In the case of affine warps, S_j and H_j are given by:

$$S_j = w_j \, [\, x_j D_{x,j} \; x_j D_{y,j} \; y_j D_{x,j} \;\; y_j D_{y,j} \; D_{x,j} \; D_{y,j} \,]^T \tag{15}$$

$$H_j = w_j \begin{bmatrix} x_j^2 & 0 & x_j y_j & 0 & x_j & 0 \\ 0 & x_j^2 & 0 & x_j y_j & 0 & x_j \\ x_j y_j & 0 & y_j^2 & 0 & y_j & 0 \\ 0 & x_j y_j & 0 & y_j^2 & 0 & y_j \\ x_j & 0 & y_j & 0 & 1 & 0 \\ 0 & x_j & 0 & y_j & 0 & 1 \end{bmatrix} \tag{16}$$

where we used $D_j = [D_{x,j}, D_{y,j}]$, $x_j = p_{x,j}$, $y_j = p_{y,j}$. Note that H_j has a lot of null or identical elements. Without any loss of information, we can thus reduce it to the vector:

$$\overline{H}_j = w_j \left[1 \; x_j \; y_j \; x_j^2 \; x_j y_j \; y_j^2 \right]^T \tag{17}$$

A message containing the two vectors $\mathbf{S}$ and $\overline{\mathbf{H}}$ is then enough to convey all the information needed to align the nodes in the new image. Notice that the messages are not expressed in the same space as the feature points or the nodes. This way, displacements can be accumulated through small 12-element messages in order to compute the affine alignment of the nodes without any loss of information. Moreover, an affine warp can be computed (given that enough nodes have been visited) with those messages while each node only needs to provide a translation. With a propagation scheme inspired from Belief Propagation [8, 9], the computation of the warp update for each node i can be summarised in 3 steps :

1. Initialise the information for each node k of the graph using equations 15 and 17 and send a first message to each neighbouring node $i \in \mathcal{N}(k)$:

$$\mathbf{S}_{ki}^{0} = S_k; \tag{18}$$

$$\overline{\mathbf{H}}_{ki}^{0} = \overline{H}_k; \tag{19}$$

2. Propagate the information between the nodes for l iterations (for a message sent from node i to node j):

$$\mathbf{S}_{ij}^{l} = S_i + \sum_{k \in \mathcal{N}(i) \setminus j} w_{k|i} \mathbf{S}_{ki}^{l-1} \tag{20}$$

$$\overline{\mathbf{H}}_{ij}^{l} = \overline{H}_i + \sum_{k \in \mathcal{N}(i) \setminus j} w_{k|i} \overline{\mathbf{H}}_{ki}^{l-1} \tag{21}$$

3. Compute the update of the warp parameters for each node j:

$$\mathbf{S}_j \;=\; S_j + \sum_{k \in \mathcal{N}(j)} w_{k|j} \mathbf{S}_{kj}^{l} \tag{22}$$

$$\overline{\mathbf{H}}_j \;=\; \overline{H}_j + \sum_{k \in \mathcal{N}(j)} w_{k|j} \overline{\mathbf{H}}_{kj}^{l} \tag{23}$$

$$\mathbf{H}_j \leftarrow \overline{\mathbf{H}}_j \tag{24}$$

$$\Delta \mathbf{p}_j \;=\; \mathbf{H}_j^{-1} \mathbf{S}_j \tag{25}$$

with $\mathcal{N}(j)$ representing the set of neighbouring nodes to node j. This solution provides a very fast propagation method that allows each node to align itself in a new image using as much information as possible provided by other feature points. Notice that no information is lost in the message passing process. This means that updating the warp parameters by Equation 7 using a template or by Equation 25 using the message passing process will give exactly the same result (given that the messages went once throught all the nodes of the template).

4.3 Message Correction

In the previous section we made the assumption that $W(\check{\mathbf{r}}_{k|j}; \mathbf{p}_j) = W(\check{\mathbf{r}}_{k|i}; \mathbf{p}_j)$ in order to obtain Equations 12 and 14. This assumption means that a node k should be expected in the same position by all the nodes belonging to the same rigid block. Even if all the nodes are indeed rigidly connected to each other, this assumption might not be correct simply because of numerical inaccuracies in the tracking result. Consider, for example, the case of tracking a set of nodes as shown in Figure 4.3. Edgels have been extracted along a line segment and tracked for a few frames. Due to some inaccuracy in the tracking, the edgels are not in a straight line anymore. If node i aligns itself using its two direct neighbours' motion information, it will be pushed up while it should actually go down (it will also be dramatically scaled down on the vertical direction in case of affine nodes). Similarly, node j will be forced to move down making it drift even further from the correct tracking result. The reason for this problem is quite simple: each node

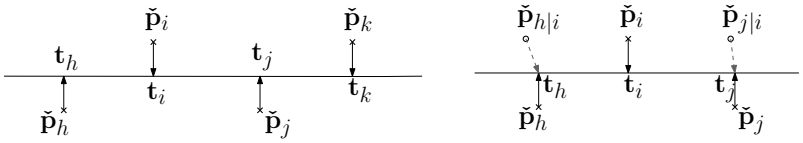

Fig. 2. Consider the edgels in Figure 2(a). Those edgels should be in straight line but are not, due to tracking inaccuracy. If nodes i and j are aligned in this new image using displacement of their direct neighbours, they will drift even further from each other while they should align along the contour. On the other hand, if they know where their neighbours should be and compute their displacement from there, the nodes will be able to correct for the current drift (see Figure 2(b) where $\check{\mathbf{p}}_{h|i} = W(\check{\mathbf{r}}_{h|i}; \mathbf{p}_i)$ and $\check{\mathbf{p}}_{j|i} = W(\check{\mathbf{r}}_{j|i}; \mathbf{p}_i)$ represent the positions nodes h and j should have with respect to node i).

acts as the model described in Section 3 but does not itself evaluate the displacement of the points used. This displacement is indeed provided by each individual node without any consideration of whether it belongs to the model of another node or not. If the assumption $W(\check{\mathbf{r}}_{k|j}; \mathbf{p}_j) = W(\check{\mathbf{r}}_{k|i}; \mathbf{p}_i)$ is verified, the points used by a model are located exactly where they are supposed to be, and the displacement information is therefore correct. If it is not verified (for numerical reason for example), the node will simply try to match this new configuration of edgels to the current image instead of trying to get back to its initial configuration. For the example of Figure 2(a), this means matching the v-shape form by h, i and j to a line segment.

By learning the correct relative positions and using them as the origin of the displacement (as shown in Figure 2(b)), the proper relative position of the nodes can be maintained and the drift problem eliminated. Concerning a message to a node i, this means that every occurrence of a position $\check{\mathbf{p}}_k = W(\check{\mathbf{r}}_{k|k} = \mathbf{0}; \mathbf{p}_k)$ must be shifted to the expected position $\check{\mathbf{p}}_{k|i} = W(\check{\mathbf{r}}_{k|i}; \mathbf{p}_i)$. By the same idea, every displacement $D_k = \mathbf{t}_k - \check{\mathbf{p}}_k$ must be replaced with the expected displacement $D_{k|i} = \mathbf{t}_k - \check{\mathbf{p}}_{k|i}$. Notice that the target $\mathbf{t}_k$ is not modified and is then only an approximation of the true target $\check{\mathbf{p}}_{k|i}$ should have provided. Indeed, the correct target $\mathbf{t}_{k|i}$ cannot be computed since the information about $\check{\mathbf{p}}_{k|i}$ is merged into $\mathbf{S}_i$ and $\overline{\mathbf{H}}_i$. However, since the drift is corrected at each frame, it is kept very small and $\mathbf{t}_k$ is therefore a good estimate of $\mathbf{t}_{k|i}$. With these modifications applied, we can see in Figure 2(b) that node i will receive coherent information, causing it to move downwards as needed.

The correction of the positions and displacements of all the points used in the alignment of a node j is a little tricky because a node j has only access to the information ($\mathbf{r}_{i|j}$and $\mathbf{p}_i$) related to its direct neighbours and the messages $m^l_{j,i} = \{\mathbf{S}^l_{ji}, \overline{\mathbf{H}}^l_{ji}\}$ they send. This means that a message coming from a neighbour i must already be corrected for i (since no spatial relation has been learnt with further points) and then adapted for j. Figure 3 shows an example of this situation where, again, we consider the case where all the points should be in a straight line while they are obviously not. So, assume that all the positions $\check{\mathbf{p}}_k$ and displacements D_k of the points k included in the message $m^l_{j,i}$ have already been corrected into $\check{\mathbf{p}}_{k|i}$ and $D_{k|i}$ respectively. In order to

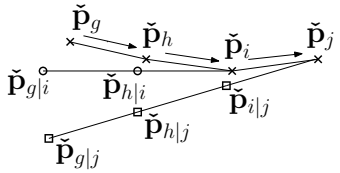

Fig. 3. Position correction: $\check{\mathbf{p}}_k$ represents the current 2D position of the edgel associated with node k, $\check{\mathbf{p}}_{k|i}$ its position as expected by i and $\check{\mathbf{p}}_{k|j}$ its position as expected by j. In this example, nodes should be align along a straight line while they are more in a curve configuration. If node i has already corrected the position of the nodes on its left into a straight line, all is left to do for node j is to apply a single warp to all the $\check{\mathbf{p}}_{k|i}$ to align them with the $\check{\mathbf{p}}_{k|j}$. The warp needed to do that is the one that aligns the affine parameters $\mathbf{p}_i$ to $\mathbf{p}_{i|j}$.

obtain the positions $\check{\mathbf{p}}_{k|j}$ expected by j, the only thing left to do is to adapt the positions $\left\{\check{\mathbf{p}}_{1|i}, \ldots, \check{\mathbf{p}}_{i|i}\right\}$ proposed by node i to the positions $\left\{\check{\mathbf{p}}_{1|j}, \ldots, \check{\mathbf{p}}_{i|j}\right\}$ expected by node j. The correction is the same for all the positions included in the message. Given that $\mathbf{p}_i$ and $\mathbf{p}_{i|j}$ are known by node j they can be used to compute the transformation needed to go from $\check{\mathbf{p}}_{k|i}$ to $\check{\mathbf{p}}_{k|j}$. This transformation is simply given by

$$\mathbf{x}_{k|j} = W(W(\check{\mathbf{p}}_{k|i}; \mathbf{p}_i^{-1}); \mathbf{p}_{i|j}) \tag{26}$$

for any $\check{\mathbf{p}}_{k|i}$. Equation 26 means that a position $\check{\mathbf{p}}_{k|i}$ is warped back into the referential of node i and then warped into the image using the warp parameters $\mathbf{p}_{i|j}$.

Since we do not have direct access to positions $\check{\mathbf{p}}_{k|i}$, we will have to apply this transformation directly to the message, i.e. to $\mathbf{S}^l_{ji}$ and $\overline{\mathbf{H}}^l_{ji}$. Using the notation

$$\begin{bmatrix} x_{k|j} \\ y_{k|j} \\ 1 \end{bmatrix} = W(W(\check{\mathbf{p}}_{k|i}; \mathbf{p}_i^{-1}); \mathbf{p}_{i|j}) = \begin{bmatrix} a & c & v \\ b & d & w \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{k|i} \\ y_{k|i} \\ 1 \end{bmatrix} \tag{27}$$

for the correction of the points in the message, we will now apply this correction directly to the Hessian part of the message corrected by node i and sent to j, i.e.

$$\overline{\mathbf{H}}^l_{ji|i} = \sum_k w_{k|i} \begin{bmatrix} 1 & x_{k|i} & y_{k|i} & x_{k|i}^2 & x_{k|i}y_{k|i} & y_{k|i}^2 \end{bmatrix}^T \tag{28}$$

The corrected Hessian part $\overline{\mathbf{H}}^l_{ji|j}$ is obtained using equation 27 on each of its terms of, which gives

$$\overline{\mathbf{H}}^l_{ji|j} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ v & a & c & 0 & 0 & 0 \\ w & b & d & 0 & 0 & 0 \\ v^2 & 2av & 2cv & a^2 & 2ac & c^2 \\ vw & aw+bv & cw+dv & ab & ad+bc & cd \\ w^2 & 2bw & 2dw & b^2 & 2bd & d^2 \end{bmatrix} \overline{\mathbf{H}}^l_{ji|i} \tag{29}$$

The correction of $\mathbf{S}_{ij}$ is somewhat more difficult because it also depends on $D_{k|i} = \mathbf{t}_k - \check{\mathbf{p}}_{k|i}$. The target position $\mathbf{t}_k$ is not modified by the correction, so we replace $D_{k|i} = [D_{x,k|i}, D_{y,k|i}]$ by $[t_{x,k} - x_{k|i}, t_{y,k} - y_{k|i}]$ in

$$\mathbf{S}^l_{ij|i} = \sum_k w_{k|i} \, [\, x_{k|i} D_{x,k|i} \; x_{k|i} D_{y,k|i} \; y_{k|i} D_{x,k|i} \;\; y_{k|i} D_{y,k|i} \; D_{x,k|i} \; D_{y,k|i} \,]^T \quad (30)$$

and apply the same correction as for $\overline{\mathbf{H}}^l_{ji|i}$ to yield

$$\mathbf{S}^l_{ij|j} = \begin{bmatrix} a & 0 & c & 0 & v & 0 \\ 0 & a & 0 & c & 0 & v \\ b & 0 & d & 0 & w & 0 \\ 0 & b & 0 & d & 0 & w \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \mathbf{S}^l_{ij|i} + \begin{bmatrix} 0 & v & 0 & a & c & 0 \\ 0 & 0 & v & 0 & a & c \\ 0 & w & 0 & b & d & 0 \\ 0 & 0 & w & 0 & b & d \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \overline{\mathbf{H}}^l_{ji|i} - \begin{bmatrix} \overline{\mathbf{H}}^l_{ji|j}(4) \\ \overline{\mathbf{H}}^l_{ji|j}(5) \\ \overline{\mathbf{H}}^l_{ji|j}(5) \\ \overline{\mathbf{H}}^l_{ji|j}(6) \\ \overline{\mathbf{H}}^l_{ji|j}(2) \\ \overline{\mathbf{H}}^l_{ji|j}(3) \end{bmatrix} \quad (31)$$

Using these two equations to correct the messages allows us to solve the drift problem by maintaining the nodes at their learnt relative positions, making the tracking more robust to occlusions and clutter.

5 Learning Spatial Relations and Weights

As we noted above, the learnt relations are key to successful tracking with Warp Propagation. Not only do they define the shape of the correlated neighbourhood used in the tracking through their weight, but they also model the expected relative configurations that keep the feature points in proper relative positions. The relative expected configuration $\mathbf{r}_{i|j}$ of a node i with respect to a node j is learnt from the observed affine parameters of i in the affine space of j. Since those relations are learnt online during tracking, they should be reliable from the first frame even with an obviously incomplete data set to be learnt from. This means that, while the relations are expected to assist in tracking, they cannot exert an overly strong bias that would hamper it. Earlier we proposed an Uncertain Potential Function that solves this problem for visual feature graph tracking with Sequential Belief Propagation by combining an informative Gaussian model learnt from the previous observations with a non-informative part [7]. This potential function for the relative position of a node i expressed in the affine space of a node j was given by

$$\psi_{i|j}(p_i, p_j) = \lambda_{i|j} \mathbf{e}^{-\frac{1}{2}(\mathbf{s}_{i|j} - \mathbf{r}_{i|j}) \tilde{\Sigma}^{-1}_{i|j} (\mathbf{s}_{i|j} - \mathbf{r}_{i|j})} + (1 - \lambda_{i|j}) \quad (32)$$

where $\mathbf{s}_{i|j}$ represents the observed parameters p_i expressed in the space defined by p_j, $\mathbf{r}_{i|j}$ is the learnt relative configuration (i.e. the mean of the relative configurations already observed), $\lambda_{i|j}$ is the probability that the model is indeed Gaussian and $\tilde{\Sigma}_{i|j}$ is a covariance matrix that accounts for the uncertainty related to the incomplete data set. Due to lack of space here, we refer the interested reader to our earlier work [7] for more details.

While this spatial relation representation is specially designed to be used for the tracking during its learning phase, it is also specific to probabilistic feature graphs, which our representation is not. Nevertheless, very few changes are needed to adapt this uncertain model to our problem. Indeed, the computed mean $\mathbf{r}_{i|j}$ already represents the most likely relative affine configuration we used in Section 4, so we simply have to compute our relational weight using the covariance matrix $\tilde{\Sigma}_{ij}$ and model probability $\lambda_{i|j}$ from the uncertain model [7]:

$$w_{i|j} = \lambda_{i|j} \prod_n \mathrm{e}^{-\frac{\tilde{\Sigma}_{nn,i|j}}{\sigma^2_{n,i|j}}} \tag{33}$$

where $\tilde{\Sigma}_{nn,i|j}$ represents the nth diagonal element of $\tilde{\Sigma}_{i|j}$, and $\sigma_{i|j} = [\sigma_{1,i|j}, \ldots, \sigma_{6,i|j}]$ defines the level of variance accepted for each parameter. This way, the more variance we observe in the relative configuration of two feature points, the less weight each one will give to a message coming from the other.

6 Experiments

In this section, we demonstrate the performance of our method on a set of representative examples of simultaneous learning and tracking. Since we are interested in articulated objects, we propose to use points extracted along their skeleton (let's call them skedgels) in addition to edgels. Those points do not have an appearance in the image and instead rely on edgels to infer their displacement. Models are initialised as shown in the first row of Figure 6 where relations are created between each pair of skedgels within a distance lower than a given threshold and between skedgels and edgels using Delaunay triangulation. The model is tracked using Warp Propagation with 10 iterations, and relations between skedgels are updated at each frame with the method explained in Section 5. Results can be found in Figure 6 where the last row represents the relation weights between skedgels at the end of the video. Notice that, while edgels and skedgels on their own would slide long the objects, here their correct position is maintained thanks to Warp Propagation. Thanks to the learnt relations, the influence neighbourhood is limited to the rigid parts.

On a Pentium Core 2 Duo 2x2 GHz with 2Gb of RAM, the tracking time for each frame (including the likelihood propagation) is between 1.6 and 3.6ms, and the learning time is between 0.3 and 0.7ms. The slowest sequence is the hand with 99 skedgels, 319 edgels and 99 relations. The fastest is the finger with 42 skedgels, 203 edgels and 41 relations.

7 Discussion

We presented a new framework for efficient propagation of alignment information through a feature-point graph. Instead of propagating potential functions as is usually done, we propagate only the motion information needed to align feature points and their

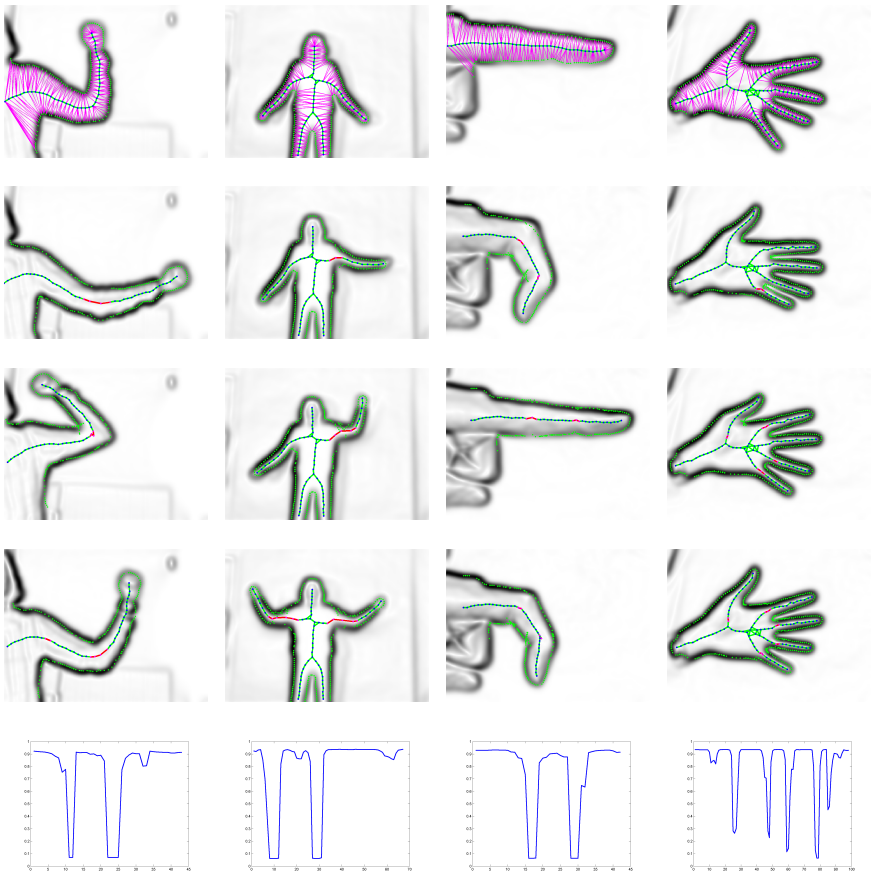

Fig. 4. Examples of simultaneous modelling and tracking of articulated objects. The first row shows the graphs as they are initialised in the first frame. The three central rows show intermediate results during the video (arm: frames 40, 90 and 255, body: 60, 130 and 330, finger: 90, 230, 310 and hand: 70, 230 and 310). Connections in red between skedgels correspond to relations with a weight lower than 0.5. The last row shows the weights of the relations in the last frame of the video. The y-axis corresponds to the weights (range between 0 and 1) and the x-axis correspond to the index of the relation. Although the correspondences of these indices to the feature graphs are not shown, it is evident that there are clean cuts in the graph with weights close to 0 that correspond to non-rigid parts, while the rigid relations have a weight close to one.

surroundings in the image. We showed that this solution allows us to simultaneously track and learn unknown, articulated objects in a few milliseconds per frame, making our solution practical in real-time scenarios even with a large number of feature points. This article focused mainly on tracking but, in the future, it would be interesting to provide a learning scheme for articulated relations instead of simple Gaussian models.

8 Acknowledgement

This work is supported by a grant from the Belgian National Fund for Research in Industry and Agriculture (FRIA) to A. Declercq and by the EU Cognitive Systems project PACO-PLUS (IST-FP6-IP-027657).

References

1. Sudderth, E.B., Mandel, M.I., Freeman, W.T., Willsky, A.S.: Visual hand tracking using nonparametric belief propagation. In: CVPRW'04 - Volume 12, Washington, DC, USA, IEEE Computer Society (2004)
2. Wu, Y., Hua, G., Yu, T.: Tracking articulated body by dynamic markov network. In: ICCV'03, Washington, DC, USA, IEEE Computer Society (2003) 1094
3. Ross, D.A., Tarlow, D., Zemel, R.S.: Unsupervised learning of skeletons from motion. In: ECCV'08, Berlin, Heidelberg, Springer-Verlag (2008) 560–573
4. Yan, J., Pollefeys, M.: Automatic kinematic chain building from feature trajectories of articulated objects. In: CVPR'06, Washington, DC, USA, IEEE Computer Society (2006) 712–719
5. Ramanan, D., Forsyth, D.A., Barnard, K.: Building models of animals from video. PAMI'06 **28** (2006) 1319–1334
6. Krahnstoever, N., Yeasin, M., Sharma, R.: Automatic acquisition and initialization of articulated models. Mach. Vision Appl. **14** (2003) 218–228
7. Declercq, A., Piater, J.H.: On-line simultaneous learning and tracking of visual feature graphs. Online Learning for Classification Workshop, CVPRW'07 (2007)
8. Yedidia, J.S., Freeman, W.T., Weiss, Y.: Understanding belief propagation and its generalizations. (2003) 239–269
9. Sudderth, E.B., Ihler, A.T., Freeman, W.T., Willsky, A.S.: Nonparametric belief propagation. In: CVPR'03. Volume 1. (2003) I–605–I–612 vol.1
10. Leordeanu, M., Collins, R.: Unsupervised learning of object features from video sequences. In: CVPR'05 - Volume 1, IEEE Computer Society (2005) 1142–1149
11. Yin, Z., Collins, R.: On-the-fly object modeling while tracking. In: CVPR'07, IEEE Computer Society (2007) 1–8
12. Drouin, S., Hébert, P., Parizeau, M.: Incremental discovery of object parts in video sequences. Comput. Vis. Image Underst. **110** (2008) 60–74
13. Briers, M., Doucet, A., Singh, S.S.: Sequential auxiliary particle belief propagation. In: Information Fusion, 2005 8th Int. Conf. on. Volume 1. (2005) 8 pp.+
14. Hua, G., Wu, Y.: Multi-scale visual tracking by sequential belief propagation. CVPR'04 **1** (2004) 826–833
15. Baker, S., Matthews, I.: Lucas-kanade 20 years on: A unifying framework. Int. J. Comput. Vision **56** (2004) 221–255
16. Lucas, B.D., Kanade, T.: An iterative image registration technique with an application to stereo vision. In: IJCAI'81. (1981) 674–679