Multivariate Relevance Vector Machines for Tracking

Arasanathan Thayananthan[1], Ramanan Navaratnam[1], Björn Stenger[2],
Philip H. S. Torr[3], and Roberto Cipolla[1]

[1] University of Cambridge, UK   {at315,rn246,cipolla}@eng.cam.ac.uk
[2] Toshiba Corporate R&D Center, Kawasaki, Japan   bjorn@cantab.net
[3] Oxford Brookes University, UK   philiptorr@brookes.ac.uk

Abstract. This paper presents a learning based approach to tracking articulated human body motion from a single camera. In order to address the problem of pose ambiguity, a one-to-many mapping from image features to state space is learned using a set of relevance vector machines, extended to handle multivariate outputs. The image features are Hausdorff matching scores obtained by matching different shape templates to the image, where the multivariate relevance vector machines (MVRVM) select a sparse set of these templates. We demonstrate that these Hausdorff features reduce the estimation error in clutter compared to shape-context histograms. The method is applied to the pose estimation problem from a single input frame, and is embedded within a probabilistic tracking framework to include temporal information. We apply the algorithm to 3D hand tracking and full human body tracking.

1 Introduction

This paper considers the problem of estimating the 3D pose of an articulated object such as the human body from a single view. This problem is difficult due to the large number of degrees of freedom and the inherent ambiguities that arise when projecting a 3D structure into the 2D image [5, 9]. In generative methods for tracking, the pose is estimated using a 3D geometric model and a likelihood function that evaluates different pose estimates. For example, various algorithms based on particle filtering have been proposed for human body or hand tracking [7, 15, 17, 26]. However, in order to track the motion of the full body or the hand, a large number of particles and a strong dynamic model are required.

More importantly, in order to build a practical system, the initialization task needs to be solved. This can be seen as an multi-object recognition problem, where recognizing a single object corresponds to recognizing the articulated object in a particular pose. Once this problem is solved, temporal information can be used to smooth motion and resolve potential pose ambiguities. This divides the continuous pose estimation task into two distinct problems: (1) estimate a distribution of possible configurations from a single frame, (2) combine frame-by-frame estimates to obtain smooth trajectories.

One approach to pose estimation is to generate a large database of examples from a 3D model and use efficient techniques to classify the current input image,
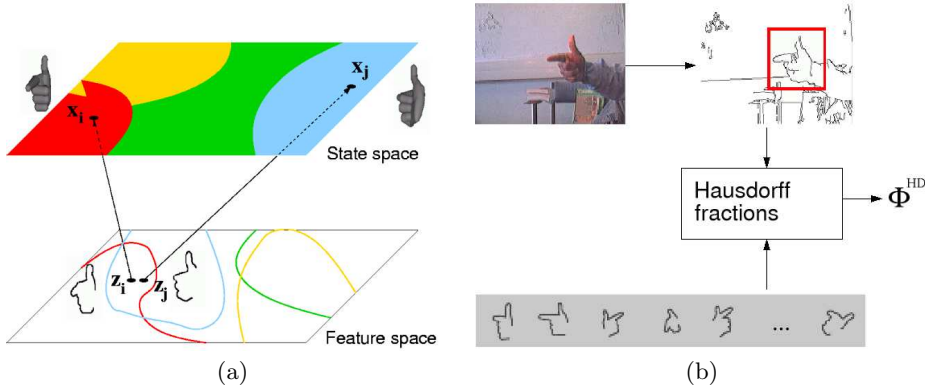
Fig. 1. (a) Multiple mapping functions. *Given a single view, the mapping from image features to pose is inherently one-to-many. Mutually exclusive regions in state space can correspond to overlapping regions in feature space. This ambiguity can be resolved by learning several mapping functions from the feature space to different regions of the state space.* **(b) Feature extraction.** *The features are obtained from matching costs (Hausdorff fractions) of shape templates to the edge map. These costs are used for creating the basis function vector ϕ^{HD}.*

e.g. using hierarchical search [18] or hashing techniques [14]. The main problem in this approach, however, is the very large number of templates required to represent the pose space. The number of templates depends on the range of possible motion and required accuracy, and can be in the order of hundreds of thousands of templates [14]. Only a fraction of the templates is searched for each query image, however all templates need to be stored.

The method for hand pose estimation from a single image by Rosales *et al.* addressed some of these issues [13]. Image features were directly mapped to likely hand poses using a set of *specialized mappings*. A 3D model was projected into the image in these hypothesized poses and evaluated using an image based cost function. The features used were low-dimensional vectors of silhouette shape moments, which are often not discriminative enough for precise pose estimation.

Agarwal and Triggs proposed a method for selecting relevant features using RVM regression [1]. The used image features were shape-contexts [4] of silhouette points. Pose estimation was formulated as a one-to-one mapping from the feature space to pose space. This mapping required about 10% of the training examples. The method was further extended to include dynamic information by joint regression with respect to two variables, the feature vector and a predicted state obtained with a dynamic model [2]. There are two concerns with this approach. Firstly, features from a single view, such as silhouettes, are often not powerful enough to solve the pose ambiguity problem. The mapping from silhouette features to state space is inherently one-to-many, as similar features can be generated by regions in the parameter space that are far apart, see figure 1(a). Hence it is important to maintain multiple hypotheses over time. The second concern is that shape-context features have been shown to be sensitive

to background clutter [20] and hence a relatively clean silhouette is needed as input. In this paper we propose the use of robust measures that are based on edge-based template matching. Edge-based matching has been used in a number of pose estimation and tracking algorithms [8, 12, 18, 23].

In this paper the pose estimation problem from template matching is formulated as learning one-to-many mapping functions that map from the feature space to the state space. The features are Hausdorff matching scores, which are obtained by matching a set of shape templates to the edge map of the input image, see figure 1(b). A set of RVM mapping functions is then learned to map these scores to different state-space regions to handle pose ambiguity, see figure 1(a). Each mapping function achieves sparsity by selecting only a small fraction of the total number of templates. However, each RVM function will select a different set of templates. This work is closely related to the work of Sminchisescu et al. [16] and Agarwal et al. [3]. Both follow a mixture of experts [11] approach to learn a number of mapping functions (or experts). A gating function is learned for each mapping function during training, and these gating functions are then used to assign the input to one or many mapping functions during the inference stage. In contrast, we use likelihood estimation from projecting the 3D-model to verify the output of each mapping function.

The main contributions of this paper are (1) an EM type algorithm for learning a one-to-many mapping using a set of RVMs, resulting in a sparse set of templates, (2) an extension of the RVM algorithm to multivariate outputs, (3) improving the robustness to image clutter using Hausdorff fractions, and (4) the application to the pose estimation problem and embedding within a probabilistic tracking framework.

The rest of the paper is organized as follows: The algorithm for learning the one-to-many mapping using multiple RVMs is introduced in section 2. Section 3 describes a scheme for training the parameters of a single RVM mapping function with multivariate outputs and section 4 explains the image features, which are based on Hausdorff matching. The pose estimation and tracking framework is presented in section 5, and results on hand tracking and full body tracking are shown in section 6. We conclude in section 7.

2   Learning multiple RVMs

The pose of an articulated object, in our case a hand or a full human body, is represented by a parameter vector $\mathbf{x} \in \mathbb{R}^M$. The features $\mathbf{z}$ are Canny edges extracted from the image. Given a set of training examples or templates $\mathcal{V} = \{v^{(n)}\}_{n=1}^N$ consisting of pairs $v^{(n)} = \{(\mathbf{x}^{(n)}, \mathbf{z}^{(n)})\}$ of state vector and feature vector, we want to learn a one-to-many mapping from feature space to state space. We do this by learning K different regression functions, which map the input $\mathbf{z}$ to different regions in state space. We choose the following model for the regression functions

$$\mathbf{x} = \mathbf{W}^k \phi(\mathbf{z}) + \boldsymbol{\xi}^k, \tag{4}$$

Algorithm 1 EM for learning multiple mapping functions $\mathbf{W}_k$

---

1. Initialize

Partition the training set $\mathcal{V}$ into K subsets by applying the K-means algorithm on the state variable $\mathbf{x}_n$ of each data point v_n. Initialize probability matrix $\mathbf{C}$.

2. Iterate

(i) Estimate regression parameters

Given the matrix $\mathbf{C} \in \mathbb{R}^{N \times K}$, where element $c_{nk} = c_k^{(n)}$ is the probability that sample point n belongs to mapping function k, learn the parameters $\left\{\mathbf{W}^k, \mathbf{S}^k\right\}$ of each mapping function, by multivariate RVM regression minimizing the following cost function

$$L^k \;=\; \sum_{n=1}^{N} c_k^{(n)} \left(\mathbf{y}_k^{(n)}\right)^T \mathbf{S}^k \left(\mathbf{y}_k^{(n)}\right), \text{ where } \mathbf{y}_k^{(n)} = x^{(n)} - \mathbf{W}^k \phi(\mathbf{z}^{(n)}). \qquad (1)$$

Note: for speed up, samples with low probabilities may be ignored.

(ii) Estimate probability matrix C

Estimate the probability of each example belonging to each of the mapping function:

$$p(\mathbf{x}^{(n)}|\mathbf{z}^{(n)}, \mathbf{W}^k, \mathbf{S}^k) = \frac{1}{2\pi|\mathbf{S}|^{1/2}} \exp\left\{-0.5 \left(\mathbf{y}_k^{(n)}\right)^T \mathbf{S}^k \left(\mathbf{y}_k^{(n)}\right)\right\}, \qquad (2)$$

$$c_k^{(n)} = \frac{p(\mathbf{x}^{(n)}|\mathbf{z}^{(n)}, \mathbf{W}^k, \mathbf{S}^k)}{\sum_{j=1}^{K} p(\mathbf{x}^{(n)}|\mathbf{z}^{(n)}, \mathbf{W}^j, \mathbf{S}^j)}. \qquad (3)$$

---

where $\boldsymbol{\xi}^k$ is a Gaussian noise vector with $\mathbf{0}$ mean and diagonal covariance matrix $\mathbf{S}^k = diag\left\{(\sigma_1^k)^2, \ldots, (\sigma_M^k)^2\right\}$. Here $\phi(\mathbf{z})$ is a vector of basis functions of the form $\phi(\mathbf{z}) \;=\; [1, G(\mathbf{z}, \mathbf{z}^{(1)}), G(\mathbf{z}, \mathbf{z}^{(2)}), ..., G(\mathbf{z}, \mathbf{z}^{(N)})]^T$, where G can be any function that compares two sets of image features. The weights of the basis functions are written in matrix form $\mathbf{W}^k \in \mathbb{R}^{M \times P}$ and $P \;=\; N + 1$. We use an EM type algorithm, outlined in Algorithm 1, to learn the parameters $\left\{\mathbf{W}^k, \mathbf{S}^k\right\}_{k=1}^{K}$ of the mapping functions. The regression results on a toy dataset are shown in figure 2.

The case of ambiguous poses means that the training set contains examples that are close or the same in feature space but are far apart in state space, see figure 1(a). When a single RVM is trained with this data, the output states tend to average different plausible poses [1]. We therefore experimentally evaluated the effect of learning mapping functions with different numbers of RVMs (with Hausdorff fractions as the input to the mapping functions, see section 4). The data was generated by random sampling from a region in the 4-dimensional state space of global rotation and scale, and projecting a 3D hand model into the image. The size of the training set was 7000 and the size of the test set was 5000. Different numbers of mapping functions were trained to obtain a one-to-many mapping from the features to the state space. The results are shown in figure 3(a). Training multiple mapping functions reduces the estimation error and creates sparser template sets. Additionally, the total training time is reduced because the RVM training time increases quadratically with the number of data points and the samples are divided among the different RVMs.
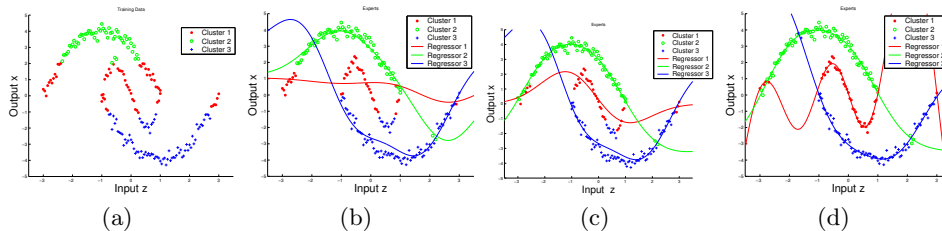
Fig. 2. RVM regression on a toy dataset. *The data set consists of 200 samples from three polynomial functions with added Gaussian noise. (a) Initial clustering using K-means. (b), (c),(d) Learned RVM regressors after the 1st, 4th and 10th iteration, respectively. Each sample data is shown with the colour of the regressor with the highest probability. A Gaussian kernel with a kernel width of 1.0 was used to create the basis functions. Only 14 samples were retained after convergence.*

# RVMs	relevant templates	approx. total training time	mean RMS error
1	13.48 %	360 min	15.82°
5	13.04 %	150 min	7.68°
10	10.76 %	90 min	5.23°
15	9.52 %	40 min	4.69°
20	7.78 %	25 min	3.89°

(a)
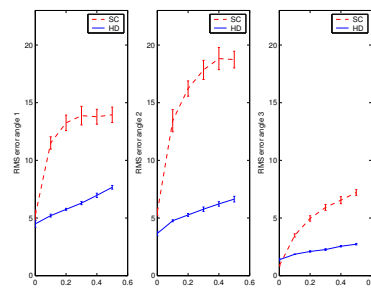
(b)

Fig. 3. (a) Single vs. multiple RVMs. *Results of training different numbers of RVMs on the same dataset. Multiple RVMs learn sparser models, require less training time and yield a smaller estimation error.*
(b) Robustness analysis. *Pose estimation error when using two different types of features: histograms of shape contexts (SC) and Hausdorff matching costs (HD). Plotted is the mean and standard deviation of the RMS error of three estimated pose parameters as a function of image noise level. Hausdorff features are more robust to edge noise.*

3 Training an RVM with multivariate outputs

During the regression stage, each mapping function is learned using an extension of the RVM regression algorithm [21]. The attraction of the RVM is that it has good generalization performance, while achieving sparsity in the representation. For our case this means that the matrices $\mathbf{W}^k$ only have few non-zero columns. Each column corresponds to the Hausdorff scores obtained by matching a specific shape template to the examples edge maps. Hence, only a fraction of the total number of shape templates needs to be stored. The RVM is a Bayesian regression framework, in which the weights of each input example are governed by a set of hyperparameters. These hyperparameters describe the posterior distribution of the weights and are estimated iteratively during training. Most hyperparameters approach infinity, causing the posterior distributions of the effectively setting the

corresponding weights to zero. The remaining examples with non-zero weights are called *relevance vectors*.

Tipping's formulation in [21] only allows regression from multivariate input to a univariate output variable. One solution is to use a single RVM for each output dimension. For example, Williams *et al.* used separate RVMs to track the four parameters of a 2D similarity transform of an image region [25]. This solution has the drawback that one needs to keep separate sets of selected examples for each RVM. We introduce the multivariate RVM (MVRVM) which extends the RVM framework to multivariate outputs, making it a general regression tool.[4] This formulation allows us to choose the same set of templates for all output dimensions.

A ridge regression scheme is used in [1, 2], which also allows selecting the same templates for all output dimensions. However, ridge regression directly optimizes over the weights without the use of hyperparameters. In contrast, we extend the framework in [21] to handle multivariate outputs. A data likelihood is obtained as a function of weight variables and hyperparameters. The weight variables are then analytically integrated out to a obtain marginal likelihood as function of the hyperparameters. An optimal set of hyperparameters is obtained by maximizing the marginal likelihood over the hyperparameters using a version of the fast marginal likelihood maximization algorithm [22]. The optimal weight matrix is obtained using the optimal set of hyperparameters.

The rest of this section details our proposed extension of the RVM framework to handle multivariate outputs and how this is used to minimize the cost function described in eqn (1) and learn the parameters of a mapping function, $\mathbf{W}^k$ and $\mathbf{S}^k$. We can rewrite eqn (1) in the following form

$$L^k = \sum_{n=1}^{N} \log \mathcal{N}(\hat{\mathbf{x}}_k^{(n)}|\mathbf{W}^k \hat{\boldsymbol{\phi}}_k(\mathbf{z}^{(n)}), \mathbf{S}^k), \tag{5}$$

$$\text{where,} \quad \hat{\mathbf{x}}_k^{(n)} = \sqrt{c_k^{(n)}}\mathbf{x}^{(n)} \quad \text{and} \quad \hat{\boldsymbol{\phi}}_k(\mathbf{z}^{(n)}) = \sqrt{c_k^{(n)}}\boldsymbol{\phi}(\mathbf{z}^{(n)}) \tag{6}$$

We need to specify a prior on the weight matrix to avoid overfitting. We follow Tipping's relevance vector approach [21] and assume a Gaussian prior for the weights of each basis function. Let $\mathbf{A} = diag(\alpha_1^{-2}, \ldots, \alpha_P^{-2})$, where each element α_j is a hyperparameter that determines the *relevance* of the associated basis function. The prior distribution over the weights is then

$$p(\mathbf{W}^k|\mathbf{A}^k) = \prod_{r=1}^{M} \prod_{j=1}^{P} \mathcal{N}(w_{rj}^k|0, \alpha_j^{-2}) , \tag{7}$$

where w_{rj}^k is the element at (r, j) of the weight matrix $\mathbf{W}^k$. We can now completely specify the parameters of the kth mapping function as $\{\mathbf{W}^k, \mathbf{S}^k, \mathbf{A}^k\}$. As the form and the learning routines of parameters of each expert are the same, we

---

[4] Code is available from *http://mi.eng.cam.ac.uk/~at315/MVRVM.htm*

drop the index k for clarity in the rest of the section. A likelihood distribution of the weight matrix $\mathbf{W}$ can be written as

$$p(\{\hat{\mathbf{x}}^{(n)}\}_{n=1}^N|\mathbf{W},\mathbf{S}) = \prod_{n=1}^N \mathcal{N}(\hat{\mathbf{x}}^{(n)}|\mathbf{W}\hat{\phi}(\mathbf{z}^{(n)}),\mathbf{S}) \ . \tag{8}$$

Let $\mathbf{w}_r$ be the weight vector for the rth component of the output vector $\mathbf{x}$, such that $\mathbf{W} = [\mathbf{w}_1,\ldots,\mathbf{w}_r,\ldots,\mathbf{w}_M]^T$ and let τ_r be the vector with the rth component of all the example output vectors. Exploiting the diagonal form of $\mathbf{S}$, the likelihood can be written as a product of separate Gaussians of the weight vectors of each output dimension:

$$p(\{\hat{\mathbf{x}}^{(n)}\}_{n=1}^N|\mathbf{W},\mathbf{S}) = \prod_{r=1}^M \mathcal{N}(\tau_r|\mathbf{w}_r\hat{\boldsymbol{\Phi}},\ \sigma_r^2) \ , \tag{9}$$

where $\hat{\boldsymbol{\Phi}} = [\mathbf{1},\hat{\phi}(\mathbf{z}_1),\hat{\phi}(\mathbf{z}_2),\ldots,\hat{\phi}(\mathbf{z}_N)]$ is the *design matrix*. The prior distribution over the weights is rewritten in the following form

$$p(\mathbf{W}|\mathbf{A}) = \prod_{r=1}^M \prod_{j=1}^P \mathcal{N}(w_{rj}|0,\alpha_j^{-2}) = \prod_{r=1}^M \mathcal{N}(\mathbf{w}_r|\mathbf{0},\mathbf{A}). \tag{10}$$

Now the posterior on $\mathbf{W}$ can be written as the product of separate Gaussians for the weight vectors of each output dimension:

$$p(\mathbf{W}|\{\hat{\mathbf{x}}\}_{n=1}^N,\mathbf{S},\mathbf{A}) \propto p(\{\hat{\mathbf{x}}\}_{n=1}^N|\mathbf{W},\mathbf{S})\,p(\mathbf{W}|\mathbf{A}) \tag{11}$$

$$\propto \quad \prod_{r=1}^M \mathcal{N}(\mathbf{w}_r|\mu_r,\boldsymbol{\Sigma}_r) \ , \tag{12}$$

where $\mu_r = \sigma_r^{-2}\boldsymbol{\Sigma}_r\boldsymbol{\Phi}^T\tau_r$ and $\boldsymbol{\Sigma}_r = (\sigma_r^{-2}\boldsymbol{\Phi}^T\boldsymbol{\Phi} + \mathbf{A})^{-1}$ are the mean and the covariance of the distribution of $\mathbf{w}_r$. Given the posterior for the weights, we can choose an optimal weight matrix if we obtain a set of hyperparameters that maximise the data likelihood in eqn (12). The Gaussian form of the distribution allows us to the remove the weight variables by analytically integrating them out. Exploiting the diagonal form of $\mathbf{S}$ and $\mathbf{A}$ once more, we marginalize the data likelihood over the weights:

$$p(\{\hat{\mathbf{x}}\}_{n=1}^N|\mathbf{A},\mathbf{S}) = \int p(\{\hat{\mathbf{x}}\}_{n=1}^N|\mathbf{W},\mathbf{S})\,p(\mathbf{W}|\mathbf{A})\,d\mathbf{W} \tag{13}$$

$$= \prod_{r=1}^M \int \mathcal{N}(\tau_r|\mathbf{w}_r\hat{\boldsymbol{\Phi}},\sigma_r^2)\,\mathcal{N}(\mathbf{w}_r|\mathbf{0},\mathbf{A}) \tag{14}$$

$$= \prod_{r=1}^M |\mathbf{H}_r|^{-\frac{1}{2}}\exp(-\frac{1}{2}\tau_r^T\mathbf{H}_r^{-1}\tau_r) \ , \tag{15}$$

where $\mathbf{H}_r = \sigma_r^2\mathbf{I} + \hat{\boldsymbol{\Phi}}\mathbf{A}^{-1}\hat{\boldsymbol{\Phi}}^T$. An optimal set of hyperparameters $\{\alpha_j^{opt}\}_{j=1}^P$ and noise parameters $\{\sigma_r^{opt}\}_{r=1}^M$ is obtained by maximising the marginal likelihood

using bottom-up basis function selection as described by Tipping et al. in [22]. Again, the method was extended to handle the multivariate outputs. Details of this extension can be found in [19]. The optimal hyperparameters are then used to obtain the optimal weight matrix:

$$\mathbf{A}^{opt} = diag(\alpha_1^{opt}, \ldots, \alpha_P^{opt}) \qquad \mathbf{\Sigma}_r^{opt} = ((\sigma_r^{opt})^{-2} \, \hat{\mathbf{\Phi}}^T \hat{\mathbf{\Phi}} + \mathbf{A}^{opt})^{-1}$$
$$\mu_r^{opt} = (\sigma_r^{opt})^{-2} \, \mathbf{\Sigma}_r^{opt} \, \mathbf{\Phi}^T \, \tau_r \qquad \mathbf{W}^{opt} = [\mu_1^{opt}, \ldots, \mu_M^{opt}]^T$$

4 Robust representation of image features

In this paper, we use Hausdorff fractions [10] in the feature comparison function G. Given two shapes represented by edge point sets $\mathbf{z}^{(i)}$ and $\mathbf{z}^{(j)}$, the Hausdorff fraction f^{HD} is defined as the ratio of points of the first shape that are within a certain distance δ from the points of the second shape:

$$f^{HD}(\mathbf{z}^{(i)}, \mathbf{z}^{(j)}) = \frac{|\mathbf{z}_\delta^{(i)}|}{|\mathbf{z}^{(i)}|}, \text{ where } \mathbf{z}_\delta^{(i)} = \{a \in \mathbf{z}^{(i)} : \min_{b \in \mathbf{z}^{(j)}} ||a - b|| < \delta\}. \qquad (16)$$

$$G^{HD}(\mathbf{z}^{(i)}, \mathbf{z}^{(j)}) = \exp\left\{-f^{HD}\right\}. \qquad (17)$$

The use of edge gradient information increases the discriminative power of these matching methods [12], thus we compute the matching cost with eight discrete orientation channels [8, 18].

We performed experiments comparing the robustness of Hausdorff fraction based features G^{HD} and features based on 100-dimensional shape-context histograms G^{SC}, described in [1, 2]. For this, a training image set is created by sampling a region in state space, in this case three rotation angles over a limited range, and using the sampled pose vectors to project a 3D hand model into the image. Because the Hausdorff features are neither translation nor scale invariant, additional training images of scaled and locally shifted examples are generated. After RVM training, a set of around 30 templates out of 200 are chosen for both, shape context and Hausdorff features. However note that the templates chosen by the RVM for each methods may differ. For testing, 200 poses are generated by randomly sampling the same region in parameter space and introducing different amounts of noise by introducing edges of varying length and curvature. Figure 3(b) shows the dependency of the RMS estimation error (mean and standard deviation) on the noise level. Hausdorff features are significantly more robust to edge noise than shape context features.

5 Pose estimation and tracking

Given a candidate object location in the image we obtain K possible poses from the mapping functions, see figure 4(a). For each mapping function $\mathbf{W}_k$ the templates selected by the RVM are matched to the input and the resulting Hausdorff fractions form the basis function vector ϕ^{HD}. We then use regression

(a) (b)
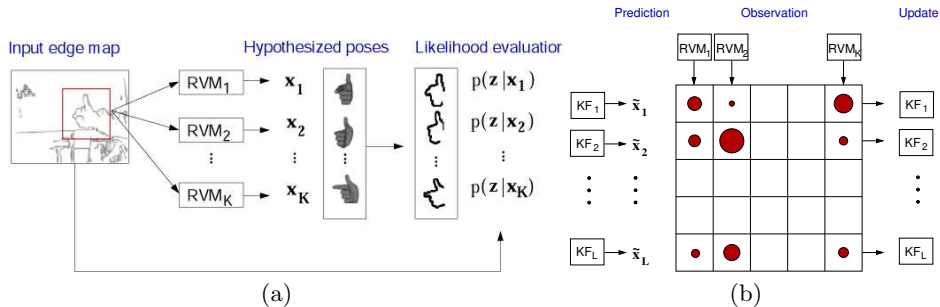
Fig. 4. (a) Pose estimation. *At each candidate location the features are obtained by Hausdorff matching and the RVMs yield pose estimates. These are used to project the 3D model and evaluate likelihoods.*
(b) Probabilistic tracking. *The modes of likelihood distribution, obtained through the RVM mapping functions, are propagated through a bank of Kalman filters [6]. The posterior distributions are represented with an L-mode piecewise Gaussian model. At each frame, the L Kalman filter predictions and K RVM observations are combined to generate possible $L \times K$ Gaussian distributions. Out of these, L Gaussians are chosen to represent the posterior probability and propagated to the next level. The circles in the figure represent the covariance of Gaussians.*

to obtain K pose estimates via $\mathbf{x}_k = \mathbf{W}^k \boldsymbol{\phi}^{HD}$. A set of candidate object locations is obtained by skin colour detection for hands and background estimation for full human body motion. Given M candidate positions we thus obtain $K \times M$ pose hypotheses, which are used to project the 3D object model into the image and obtain image likelihoods.

The observation model for the likelihood computation is based on edge and silhouette cues. As a likelihood model for hand tracking we use the function proposed in [18], which combines chamfer matching with foreground silhouette matching, where the foreground is found by skin colour segmentation. The same likelihood function is used in the full body tracking experiments, with the difference that in this case the foreground silhouette is estimated by background subtraction.

Temporal information is needed to resolve the ambiguous poses and to obtain a smooth trajectory through the state-space after the pose estimation is done at every frame. We embed pose estimation with multiple RVMs within a probabilistic tracking framework, which involves representing and maintaining distributions of the state $\mathbf{x}$ over time.

The distributions are represented using a piecewise Gaussian model [6] with L components. The evaluation of the distribution at one time instant t involves the following steps (see figure 4(b)):

(1) Predict each of the L components,
(2) perform RVM regression to obtain K hypotheses,
(3) evaluate likelihood computation for each hypothesis,

(4) compute the posterior distribution for each of $L \times K$ components,

(5) select L components to propagate to next time step,

The dynamics are modeled using a constant velocity model with large process noise [6], where the noise variance is set to the variance of the mapping error estimated at the RVM learning stage. At step (5) k-means clustering is used to identify the main components of the posterior distribution in the state space, similar to [24]. Components with the largest posterior probability are chosen from each cluster in turn, ensuring that not all components represent only one region of the state-space.

For a given frame the correct pose does not always have the largest posterior probability. Additionally, the uncertainty of pose estimation is larger in some regions in state space than in others, and a certain number of frames may be needed before the pose ambiguity can be resolved. The largest peak of the posterior fluctuates among different trajectories as the distribution is propagated. Hence a history of the peaks of the posterior probability needs to be considered before a consistent trajectory is found that links the peaks over time. In our experiments a batch Viterbi algorithm is used to find such a path.

6   Results and Evaluation

Global pose: In our first experiment, we estimate the three rotation angles and the scale of a pointing hand. We use 10 RVMs to learn the mapping. First 5000 templates are created from a 3D model by random sampling from the state-space. The task is to choose the relevant templates for pose estimation from these templates. Even though we do not estimate image plane translation using the mapping functions, we allow random translation within 7 pixels range in the generated images to achieve translation invariance within a short range. After training the RVMs, a total of 325 relevant templates out of 5000 were selected. For comparison, Stenger *et al.* used approximately 12 000 templates to estimate a similar type of motion [18]. The learned RVM mapping functions are used to estimate the rotation angles and the scale of a pointing hand in a sequence of 1100 frames. Skin colour detection is used to find candidate locations for applying the mapping functions. However, the mapping functions themselves only receive an edge map as their input. The tracking framework described in section 5 is then applied to the detection results at every frame. Figure (5) shows some example frames from this sequence.

Hand articulation : The method is applied to the hand open-close sequence with 88 frames from [18], where approximately 30 000 templates were required for tracking. To capture typical hand motion data, we use a large set of 10 dimensional joint angle data obtained from a data glove. The pose data was approximated by the first four principal components. We then projected original hand glove data into those 4 dimensions. The global motion of the hand in that sequence was limited to a certain region of the global space (80°, 60°and 40° in rotation angles and 0.6 to 0.8 in scale). The eight-dimensional state space is defined by the four global and four articulation parameters. A set of 10 000

templates is generated by random sampling in this state space. After training 10 RVMs, 455 templates out of 10 000 are retained. Due to the large amount of background clutter in the sequence, skin colour detection is used in this sequence to remove some of the background edges for this sequence. Tracking results are shown in figure (6).

Full body articulation: In order to track full body motion, we use a data set from the CMU motion capture database of walking persons ($\sim$ 9000 data points). In order to reduce the RVM training time, the data is projected onto the first six principal components.

The first input sequence is a person walking fronto parallel to the camera. The global motion is mainly limited to translation. The eight-dimensional state-space is defined by two global and six articulation parameters. A set of 13,000 training samples were created by sampling the region. We use 4 RVM mapping functions to approximate the one-to-many mapping. A set of 118 relevant templates is retained after training. Background subtraction is used to remove some of the background edges. The tracking results are shown in figure (7). The second input sequence is a video of a person walking in a circle from [15]. The range of global motion is set to 360° around axis normal to the ground plane and 20° in the tilt angle. The range of scales is 0.3 to 0.7. The nine-dimensional state-space region is defined by these three global and six articulation parameters. A set of 50 000 templates is generated by sampling this region. We use 50 RVM mapping functions to approximate the one-to-many mapping. A set of 984 relevant templates is retained after training. Background subtraction is used to remove some of the background edges. The tracking results are shown in figure (8).

Computation time: The execution time in the experiments varies from 5 to 20 seconds per frame (on a Pentium IV, 2.1 GHz PC), depending on the number of candidate locations in each frame. The computational bottleneck is the model projection in order to compute the likelihoods (approximately 100 per second). For example, for 30 search locations and 50 RVM mapping functions result in 1500 model projections, requiring 15 seconds. It can be observed that most mapping functions do not yield high likelihoods, thus identifying them early will help to reduce the computation time.

7 Summary and conclusion

This paper has introduced an EM type algorithm to learn a one-to-many mapping using multiple relevance vector machines. To this end the original RVM formulation was extended to allow for multivariate outputs. The method was applied to the problem of pose estimation from a single frame, where the RVMs were used to select relevant templates from a large set of candidate templates.

Pose estimation was embedded within a tracking framework, combining both discriminative and generative methods: At each frame the set of mappings from feature to parameter space generates a set of pose hypotheses, which are then used to project a 3D model and compute an image likelihood. The state posterior
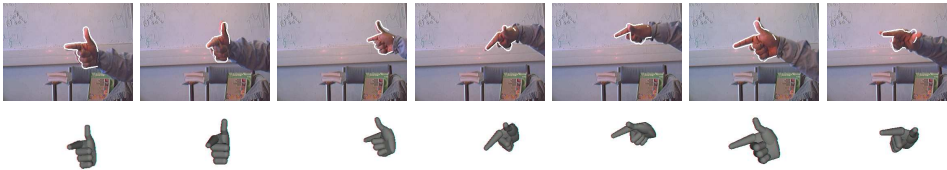
Fig. 5. Tracking a pointing hand. *Example frames from tracking a pointing hand sequence with 1100 frames using a single camera are shown. The model contours corresponding to the optimal path through the state distribution are superimposed, and the 3D model is shown below. A total of 389 relevant templates, divided between 10 RVM mapping functions, were used to estimate the hand pose. For comparison, Stenger et al. [18] used 12 000 templates to estimate a similar type of motion.*
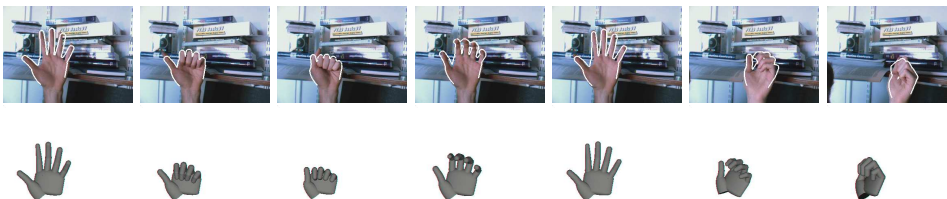
Fig. 6. Tracking an opening and closing hand. *This sequence shows tracking of opening and closing hand motion together with global motion on a sequence from [18]. A total of 537 relevant templates were used with 20 RVM mapping functions for pose estimation. As a comparison [18] used about 30 000 templates to track the same sequence.*

distribution, represented by a piecewise Gaussian distribution, is propagated over time, and dynamic information is included using a bank of Kalman filters. A batch Viterbi algorithm is used to find a path through the peaks of this distribution in order to resolve ambiguous poses.

Template-based pose estimation schemes solve the problem of initialisation and pose-recovery and maintain multiple hypothesis in tracking articulated objects. Furthermore edge-based schemes are resistant to background clutter and image deformations to a certain degree. However, a major problem is the large number of templates that are needed for the pose estimation of articulated objects [18]. We have presented a scheme where we achieve reduction of two to three orders of magnitude in the number of templates.

References

1. A. Agarwal and B. Triggs. 3D human pose from silhouettes by relevance vector regression. In *Proc. Conf. Computer Vision and Pattern Recognition*, volume II, pages 882–888, Washington, DC, July 2004.
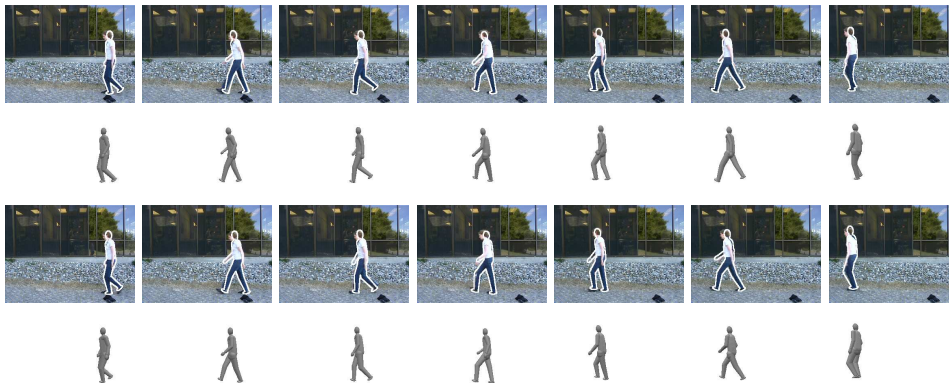
Fig. 7. Tracking a person walking fronto parallel to the camera . *The first and second rows shows the frames from [15], overlaid with the body pose corresponding to the optimal path through the posterior distribution and the corresponding the 3D model, respectively. Similarly, second and third rows show the second best path. Notice that the second path describes the walk equally well except for the right-left leg flip which is one of the common ambiguity that arises in human pose estimation from monocular view. A total of 118 templates with 4 RVM mapping functions were used.*

Fig. 8. Tracking a person walking in a circle. *This figure shows the results of the tracking algorithm on a sequence from [15]. Overlaid is the body pose corresponding to the optimal path through the posterior distribution, the 3D model is shown below. A total of 1429 templates with 50 RVM mapping functions were used.*

2. A. Agarwal and B. Triggs. Learning to track 3D human motion from silhouettes. In *In Proceedings of the 21st International Conference on Machine Learning*, pages 9–16, Banff, Canada, 2004.
3. A. Agarwal and B. Triggs. Monocular human motion capture with a mixture of regressors. In *In IEEE Workshop on Vision for Human Computer Interaction*, 2005.
4. S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Trans. Pattern Analysis and Machine Intell.*, 24(4):509–522, April 2002.
5. M. Brand. Shadow puppetry. In *Proc. 7th Int. Conf. on Computer Vision*, volume II, pages 1237–1244, Corfu, Greece, September 1999.
6. T. J. Cham and J. M. Rehg. A multiple hypothesis approach to figure tracking. In *Proc. Conf. Computer Vision and Pattern Recognition*, volume II, pages 239–245, Fort Collins, CO, June 1999.
7. J. Deutscher, A. Blake, and I. Reid. Articulated body motion capture by annealed particle filtering. In *Proc. Conf. Computer Vision and Pattern Recognition*, volume II, pages 126–133, Hilton Head, SC, June 2000.

8. D. M. Gavrila. Pedestrian detection from a moving vehicle. In *Proc. 6th European Conf. on Computer Vision*, volume II, pages 37–49, Dublin, Ireland, June/July 2000.

9. N. R. Howe, M. E. Leventon, and W. T. Freeman. Bayesian reconstruction of 3D human motion from single-camera video. In *Adv. Neural Information Processing Systems*, pages 820–826, Denver, CO, November 1999.

10. D. P. Huttenlocher, J. J. Noh, and W. J. Rucklidge. Tracking non-rigid objects in complex scenes. In *Proc. 4th Int. Conf. on Computer Vision*, pages 93–101, Berlin, May 1993.

11. M. Jordan and R. Jacobs. Hierarchical mixtures of experts and the em algorithm. *Neural Computation*, 6:181–214, 1994.

12. C. F. Olson and D. P. Huttenlocher. Automatic target recognition by matching oriented edge pixels. *Transactions on Image Processing*, 6(1):103–113, January 1997.

13. R. Rosales, V. Athitsos, L. Sigal, and S. Scarloff. 3D hand pose reconstruction using specialized mappings. In *Proc. 8th Int. Conf. on Computer Vision*, volume I, pages 378–385, Vancouver, Canada, July 2001.

14. G. Shakhnarovich, P. Viola, and T. Darrell. Fast pose estimation with parameter-sensitive hashing. In *Proc. 9th Int. Conf. on Computer Vision*, volume II, pages 750–757, 2003.

15. H. Sidenbladh, F.D.L Torre, and M. J. Black. A framework for modeling the appearance of 3d articulated figures. In *IEEE International Conference on Automatic Face and Gesture Recognition*, pages 368–375, Grenoble, France, 2000.

16. C. Sminchisescu, A. Kanaujia, Z. Li, and D. Metaxas. Discriminative density propagation for 3d human motion estimation. In *Proc. Conf. Computer Vision and Pattern Recognition*, pages 217–323, June 2005.

17. C. Sminchisescu and B. Triggs. Estimating articulated human motion with covariance scaled sampling. *Int. Journal of Robotics Research*, 22(6):371–393, 2003.

18. B. Stenger, A. Thayananthan, P. H. S. Torr, and R. Cipolla. Filtering using a tree-based estimator. In *Proc. 9th Int. Conf. on Computer Vision*, volume II, pages 1063–1070, 2003.

19. A. Thayananthan. *Template-based pose estimation and tracking of 3D hand motion*. PhD thesis, University of Cambridge, UK, 2005.

20. A. Thayananthan, B. Stenger, P. H. S. Torr, and R. Cipolla. Shape context and chamfer matching in cluttered scenes. In *Proc. Conf. Computer Vision and Pattern Recognition*, volume I, pages 127–133, 2003.

21. M. E. Tipping. Sparse Bayesian learning and the relevance vector machine. *J. Machine Learning Research*, pages 211–244, 2001.

22. M. E. Tipping and A. Faul. Fast marginal likelihood maximisation for sparse bayesian models. In *Proc. Ninth Intl. Workshop on Artificial Intelligence and Statistics*, Key West, FL, January 2003.

23. K. Toyama and A. Blake. Probabilistic tracking with exemplars in a metric space. *Int. Journal of Computer Vision*, 48(1):9–19, June 2002.

24. J. Vermaak, A. Doucet, and P. Pérez. Maintaining multi-modality through mixture tracking. In *Proc. 9th Int. Conf. on Computer Vision*, 2003.

25. O. Williams, A. Blake, and R. Cipolla. A sparse probabilistic learning algorithm for real-time tracking. In *Proc. 9th Int. Conf. on Computer Vision*, volume I, pages 353–360, Nice, France, October 2003.

26. Y. Wu, J. Y. Lin, and T. S. Huang. Capturing natural hand articulation. In *Proc. 8th Int. Conf. on Computer Vision*, volume II, pages 426–432, Vancouver, Canada, July 2001.