Regression-based hand pose estimation from multiple cameras

In this chapter an RVM-based learning method is developed for hand pose recovery. The method is based on that proposed by Agarwal and Triggs for whole body pose recovery. However, hand pose recovery appears a more challenging problem than whole body pose estimation due to the greater degree of actual occlusion, and the greater degree of apparent occlusion where finger bounding contours are lost. Further difficulties arise from the acyclic character of usual hand motions that tend to be fast and sudden in images. Also, no a priori positions can be assumed for hands. But we can use the facts that bare hands’ textures are fairly uniform and their colour has a relatively small variance in comparison to clothing. The key development proposed here is a combination of multiple views. Such method allows the use a new modification of shape contexts for rotation invariance, reducing the number of required training samples for pose estimation. An experimental comparison of the pose recovery performance using single versus multiple views is reported for synthetic and real imagery. The effects of the number of image measurements and the number of training samples on performance are also taken into account for the comparison.

7.1 Introduction

As mentioned earlier in this thesis, two quite different approaches to the problem of pose estimation of articulated objects are apparent in the literature. The first, and more traditional, is the generative
approach, in which an estimate of the pose is used to update the model that predicts the appearance, e.g. by projecting a 3D model into the image. Measurements of the deviation between prediction and reality are used to estimate the pose update. The tracking methods studied in chapters 4, 5 and 6 are generative trackers. Such trackers can achieve good qualitative pose estimates at high frame rate, but they need to rely on models that give a good approximation of the tracked object. Furthermore, these trackers need a good estimate of the initial state, and at any time, if its prediction is not a good match to the true state, tracking will fail.

The approach of discriminative algorithms has recently been more widely explored for articulated objects \[\text{AS02, AASK04, ISTC03, STTC03, Bra99, SVD03, AT04c}\]. The idea is to recover a direct, but non-physically based, mapping between a (robust) representation of appearance and the model parameters such as joint angles. Inter alia, the approach exploits, as Wu et al. \[\text{WLH01}\] note, the fact that the range of typically explored poses of hands is much smaller than the entire range.

As mentioned in Chapter 2, two main approaches to relating image measurements qualitatively to 3D poses: classification-based and mapping-based. The former is computationally expensive and can only output pose estimations that are in the training set. The second is fast and is able to output estimates in a continuous manifold of the parameters space. The key factor of mapping-based approaches is the model used to build the map.

In this chapter, a mapping-based approach is developed for the problem of hand pose estimation. This is based on a multivariate regression method that follows in part Agarwal and Triggs’ work on whole body pose estimation. However, the hand pose recovery is in general a more difficult problem, not least because of the far greater degree of actual occlusion, and of “apparent occlusion” where finger bounding contours are lost. For this reason this chapter proposes an extension of the single view method to multiple cameras, an approach which Erol et al. \[\text{EBN+05}\] points out has not been widely explored for this problem. An experimental comparison of single and multiple view performance is presented, taking into account variation in the number of image measurements and training samples needed.

A framework of the method described in this chapter is illustrated in Figure 7.1. Once the images are acquired, a pre-processing step is to extract image descriptors is performed on both the training and testing phase. This step, described in Section 7.2 produces a compact description of appearance, for
7.2 Extracting Multiple View Image Descriptors

The initial step of the method (both in training and application phases) is the conversion of each image of a hand into a silhouette contour, and thence into a compact description using shape contexts. Because of the wide variation in scale and orientation of hands in imagery, it is important to incorporate invariance to these transformations within the context. A novel modification for rotation invariance is proposed. Its

which we use, as did Agarwal and Triggs, shape contexts descriptors. A novel modification is introduced for rotation invariance without loss of information about the shape. A compact global image descriptor is obtained though vector quantisation, and multiple view information is combined by concatenation. To train the regressor, it is necessary to gather a set of training pairs of multiple view images and 3D poses, section 7.3 describes the acquisition of training and testing data. The regression method used to learn the mapping between appearance and 3D poses is described in Section 7.4. Section 7.5 describes the experiments and results. A summary and the conclusions are drawn in Section 7.6.

Figure 7.1: Framework showing the combination of the methods employed in the training phase. For each image, first the shape contexts of the silhouette contour are calculated. Next, vector quantisation is performed to produce a compact global image descriptor. The multiple view data are combined by concatenation and an RVM-based regressor is trained using the 3D poses.
description is followed by the description of our method for combination of multiple view information.

### 7.2.1 Shape Contexts

Shape contexts, proposed by Belongie et al. [BMP02], are rich shape descriptors that are usually computed for points on the silhouette contours. They encode local information about each point relative to its neighbours, and they can be made scale and rotation invariant.

Among modifications of shape contexts found in the literature, are (i) that of Ohashi and Shimodaira [OS03a, OS03b, FTR+04], which is simpler than Belongie et al.'s method, but the final image descriptor is similar to that obtained after vector quantisation (as done by Agarwal and Triggs); and (ii) that of Thayananthan et al. [TSTC03], who used edge orientation and a continuity constraint for shape context matching, so neighbouring pixels in the image have to match neighbouring pixels in the shape contexts space), but the basic image descriptor is the same. The method presented here aims to obtain a robust global image descriptor, rather than to provide a match between sets of object points.

![Figure 7.2: An example of a hand image with a cluttered background (a) and its pixel-wise silhouette contour extracted by skin detection followed by edge detection (b).](image)

Recovery of the silhouette of the hand, assumed un-gloved, is achieved using the histogram-based classifier presented in Chapter 3. This is applied to subsampled to $90 \times 120$ pixels to reduce computation cost. In our database, hands occupied about 20% ($\pm 6.2\%$ STD) of the image pixels. The shape contexts are computed only from positions on the silhouette contour, which is easily derived by edge detection in the resulting binary skin/not-skin image. Figure 7.2 shows the extraction of silhouette contour points from a hand image with clutter in the background. Note that only a few points are located outside the contour of the silhouette of the hand.

At any point on the contour, neighbouring contour points are accumulated in 60 bins arranged in
7.2 Extracting Multiple View Image Descriptors

Figure 7.3: The shape context of a point at the tip of the index finger: (a) the regions taken into account for computation; (b) the obtained context. The column order in the histogram follows a counter-clockwise scan starting from fiducial “0°” dashed line, and the row order follows from outer to inner sectors.

log-polar fashion, five along the radial direction and twelve around the polar angle, spaced equally in log-distance and angle, respectively. To provide a first layer of scale invariance, the inner radius is set proportional to the mean $\mu$ of the distances between all the pairs of points in the silhouette. In our implementation, the inner radius is $\mu/8$, and radius increases in octaves to $2\mu$, typically covering all of the hand silhouette. Figure 7.3 illustrates the construction of a shape context for a point in the silhouette of the hand shown in Figure 7.2. The resulting 60-bin histogram is normalised, providing again for scale invariance. For image $i$ the complete image description is generated as the set of $n_i$ 60-bin histograms computed at $n_i$ points along the silhouette contour.

Figure 7.4 shows the complete set of shape contexts for one hand silhouette. The $n$ points in 60 dimensions are projected onto the first two principal components. Because the individual shape contexts computed at neighbouring points do not change drastically, and the primary principal components pick out a principal plane, it is possible, even in this feature space, to discern the characteristic four fingers and thumb.

7.2.2 Rotation Invariance

As mentioned before, the orientation of hands in natural actions can largely vary. This can be a challenge for discriminative methods if each different orientation requires a new set of training samples. The solution present in this chapter is to use rotation invariant image descriptors and multiple view. Therefore
7.2 Extracting Multiple View Image Descriptors

Figure 7.4: The 60-d shape context manifold obtained from a hand silhouette, visualised via a projection onto the first two principal components.

the pose estimation can be focused on internal parameters (joint angles), and the global pose can be estimated with the use of triangulation. Another benefit of rotation invariance is that it alleviates the need for the cameras setup consistency between training and testing phases.

Essentially, rotation invariance is achieved by orienting the fiducial line of the shape contexts according to some local feature of the shape. For instance, Figure 7.5 shows the shape contexts of a point at the tip of the middle finger for different images. In this example, the fiducial line is oriented by the tangent of the local contour. Note that the difference between the shape contexts of that point in a synthetic hand image (panel a), a real image (panel b), and the same image rotated (panel c) is very small in comparison to the shape context of that point in a hand at a different grasping pose (panel d).

Belongie et al. ensured rotational invariance by aligning the fiducial “0°” line with the tangent to the silhouette contour at each point. While this works well if the contour is smooth, which in our experience requires either large images or using proper linked edge detection, the result in low resolution images, and using pixel contour points, was found to be noisy. A more robust alternative was found to be to use the geometric centre of the silhouette and to set the fiducial line to be orthogonal to the line from the centre to the contour point.

The rotation invariance of both tangent-based and centroid-based methods is obtained at the cost of
Figure 7.5: The silhouettes, log-polar bins, and the resulting shape context vectors obtained from the tip of the middle finger in four different images using tangent-oriented rotation invariant shape contexts. Context (a) is from a synthetic training image. Its similarity with real images is shown in panel (b), and (c) illustrates rotation invariance. The shape context from a different hand pose is shown in (d).

Reducing the amount of global information about the shape of the silhouettes. This can be visualised in the projection of the 60 dimensional shape contexts space shown in Figure 7.6 (b, d, and f) – note that these shape context manifolds do not present discerning hand characteristics – and through the nearest neighbour classification results shown in Figs. 7.7 and 7.8. The solution that we adopt in this chapter is to orient the shape context with the axis that links the wrist to the tip of the hand. For simplicity, we assume that two points of the silhouette contour lie on the image borders, and these points are taken to be either side of the forearm. This is more robust than, for instance, using the principal axis obtained through PCA, as it can vary abruptly depending on the hand pose. The illustrations of Figure 7.6 (g and h) and the results in Figs. 7.7 and 7.8 show that this maintains the discrimination power of non-rotation invariant shape contexts and adds robustness to planar rotations.

\[1\] The results in Figs. 7.7 and 7.8 were obtained through classification using global multiple view descriptors for the images, as described later, in Secs. 7.2.3 and 7.2.4.
Figure 7.6: Methods to orient the shape contexts for rotation invariance (left) and their respective descriptors in the 60 dimensional space of shape contexts projected in 2 dimensions using PCA: (a and b) without rotation invariance, i.e., using a fixed orientation for the whole image; (c and d) using tangents obtained with a $3 \times 3$ window (as suggested in [BMP02]); (e and f) using the orientation orthogonal to the ray from the mass centre (indicated by the blue circle); (g and h) aligning the shape contexts with the hand's axis.
7.2 Extracting Multiple View Image Descriptors

Figure 7.7: Nearest-neighbour classification results using multiple view descriptors obtained from the silhouettes on the first column, using (i) not invariant shape contexts and three methods of rotation invariance: (ii) tangent-based, (iii) centroid-oriented and (iv) hand axis-oriented shape contexts. Note that using the hand axis (iv), the 'fingers ambiguity' is avoided.

Figure 7.8: Same as Figure 7.7 for another set of testing images. Here, even though the hand is roughly aligned with the training data, tangent-based and hand axis-based rotation invariant shape contexts provided better results than the shape contexts without rotation invariance.
7.2.3 Encoding a Global Image Descriptor

In order to reduce the dimensionality needed to describe an image, a coding method is used. In this chapter, we adopt the same method as Agarwal and Triggs \[AT04a\]: vector quantisation. In the training phase, a codebook is created in a similar fashion to that of histogram, with bins being calculated using a clustering method. From a training set of \(I\) images all \(\left(\sum_{i=1}^{I} n_i\right)\) 60-d shape context vectors are clustered into \(K\) centroids using the \(K\)-means algorithm \[DH73\]. Each individual shape context \(j\) in image \(i\) becomes re-expressed as a \(K\)-dimensional vector \(x_{ij}\) with \(K - 1\) zero elements and a single unit element. In both training and application phases, the complete image descriptor \(x_i\) is generated by summing these and normalising by \(n_i\):

\[
x_i = \sum_{j=1}^{n_i} x_{ij} / n_i.
\]

To soften the effects of spatial quantisation, the descriptors are built allowing context vectors to vote with Gaussian weights into the few centres nearest to them \[AT04a\].

Figure 7.9 shows the centroids obtained from 546 training images with a total of 128659 shape context descriptors.

7.2.4 Combining Multiple View Information

Some possibilities have been considered to combine multiple view information. In \[UMKR96\], the view which is most perpendicular to the hand palm is selected and the other are discarded. However, in many cases it is difficult to estimate the orientation of the hand if each view is analysed individually.

The low level approach is to group all the shape contexts from all the images together before performing clustering to build the codebook. The problem of this approach is that the improvement obtained by using multiple views may not be very significant, as one set of measurements can be associated to more than one global orientation.

An alternative is to estimate the pose from each view individually and combine the results at a high level using, for example, a graphical model \[Mur02\]. If global pose parameters can be estimated using triangulation and if regressors can be trained with a set that is comprehensive in terms of the internal pose parameters and orientations, then the same regressor could be applied for all the cameras, and the setup
of cameras would not need to be the same as in training. However, as discussed later, it is not realistic to use very large training sets.

In [HSS02], it was demonstrated that the discrimination power is proportional to a measure of the complexity of the curvature of the contour. Thus, for each view, the matching score is weighted by this measure and they are added up to the final score. In the approach proposed here, it is not necessary to employ a shape complexity measure to weight each view: the regressor does that implicitly if linear kernels are used. For that, the information is combined in an intermediate level, by generating vectors $x$ for each camera individually and concatenating them into a higher dimensional vector that describes the current measurements from all the cameras. The regressor is then trained using these concatenated vectors, as illustrated in figure 7.1. The advantage of this approach is that the images descriptor encode information from all the views separately, reducing the number of training data needed to use the additional pose constraints that multiple views offer. The drawback is the need for an agreement of the cameras poses between training and application phases, though the descriptor is robust to rotations on
the cameras planes and to variation in scale, i.e., the proximity between the cameras and the hand can vary, as well as the internal camera parameters (e.g. focal length).

In the present implementation, $K$ is set to 30 for each view, so the image descriptors of a three-views data set are represented in a 90-dimensional manifold as shown in Figure 7.40. For comparison, 90-d single view image descriptors were also obtained from the same training set by using $K = 90$.

Note that, for multiple views, the first and second principal components are roughly aligned with the variation in $\theta_Z$, and with the overall degree of flexion of the fingers, respectively, where $\theta_Z$ is the rotation around the forearm axis, as detailed in Sec. 7.3. This hints that this dataset of hand appearances can roughly be represented with two degrees of freedom. This effect cannot be observed for single view descriptors $x$. In that case, the manifold seems to need at least three dimensions to show more separability between hands poses.

7.3 Obtaining and testing the training data

So far, we have described the generation of a possibly multiview image descriptor $x$. An essential input to the later regression process is, of course, the association of each $x$ with a set of known joint angles $y$.

For this chapter we use a subset of the hand trajectory database prepared by Thayananthan and Stenger [STTC03] at Cambridge University’s Department of Engineering, using an Immersion Corporation’s CyberGlove. The database contains the trajectory of 20 joint angles of the hand of two users. The 28 DOF hand model described in Section 5.5 was used to synthesize images at the poses of this database, so $y \in \mathbb{R}^{28}$. Since the CyberGlove used does not include sensors between the forearm and the palm, the two degrees of freedom of this joint were set to constant values.

7.3.1 Training Sets

We demonstrate experiments using two training sets. The first, dubbed open-close, consists of a trajectory that starts with all the fingers stretched after which a grasping gesture is performed in 78 frames. The glove used to generate this data did not have a global position and orientation sensor, so the trajectory was duplicated seven times for $15^\circ$ spaced values $0^\circ \leq \theta_Z \leq 90^\circ$, giving a total of 546 poses, some of which are shown in Figure 7.11. For desktop tasks the variation of the other orientation parameters ($\theta_X$ and $\theta_Y$) is usually small enough to enable us to rely on the invariance properties of the modified shape contexts.
Figure 7.10: 90-d manifolds of \( x \) vectors obtained from a training data visualised using projection onto the first two principal components. The silhouettes of the hand at some key poses (6 poses for each \( \theta_Z \) angle) are shown in their location in the manifold. Panel (a) shows the manifold for single view \( x \) vectors, and panel (b) shows the same for multiple view \( x \).
A more accurate global orientation can be obtained by triangulation when multiple views are used. For a fair comparison between single and multiple view, \( \theta_X \) and \( \theta_Y \) are not taken into account. For multicamera application, the hands were rendered from three different viewpoints. These viewpoints have the same camera calibration parameters as those used on the acquisition of real images for recognition.

The second training set, dubbed complex, has fingers moving individually, as shown in Figure 7.12. This sequence has 239 internal poses that, as before, are reproduced for 15° spaced values \( 0° \leq \theta_Z \leq 90° \), giving a total of 1673 three-dimensional poses.
7.3 Obtaining and testing the training data

7.3.2 Assessing the training set

Figure 7.13(a) shows the dissimilarity \( D \) between the \( x \) image descriptors in matrix form. The natural dissimilarity measure for histograms is the \( \chi^2 \) test statistic [BMP02]:

\[
D_{i,j} = D(x_i, x_j) = \frac{1}{2} \sum_{k=1}^{K} \frac{[x_i(k) - x_j(k)]^2}{x_i(k) + x_j(k)},
\]

(7.1)

where \( i \) and \( j \) are sample (i.e., image) indexes and \( k \) is the descriptors’ dimension index\(^2\). The dissimilarity between 28 dimensional \( y \) vectors of pose is shown in Figure 7.13(b), obtained using the Euclidean distance. The reduced number of low values in the off-diagonal elements of \( D(x_i, x_j) \) shows the discrimination power of the image descriptor. Due to the similarity of the silhouettes, there remains more confusion amongst \( x \) vectors as the fingers are closed up.

In order to assess the discriminatory power of the image descriptors \( x \), a nearest neighbour classification experiment was performed with 36 hand images – 9 hand poses taken from 4 orientations. The results, shown in Figures 7.14 and 7.15, suggest that the image descriptor is robust enough to provide a good qualitative description of the hand shape from images that are not in the training set, even though the hand model is not accurate. Figure 7.15 also shows that the use of multiple views can improve the

\(^2\)Since shape contexts are histograms, this measure was also used previously in the criterion function of the clustering algorithm for vector quantisation – Section 7.2.3.
7.3 Obtaining and testing the training data

nearest neighbour classification result.

![Distance map for image descriptors](image1)

![Input test image](image2)

Figure 7.14: (a) Distance map between the 36 testing samples (9 images with 4 different rotations) and the 546 training samples and a single view. The repetition in the pattern at each 9 samples confirms rotation invariance. (b) Nearest-neighbour classification results for nine samples of the same orientation.

![Sample images](image3)

Figure 7.15: 1\textsuperscript{st} row: sample images from camera 2 with modifications in orientation, translation and scale. The nearest-neighbour classification results using single view with scale and rotation invariant descriptors are shown in the 2\textsuperscript{nd} row. The 3\textsuperscript{rd} row shows the same, using multiple views.

An improvement is expected to be achieved with regression because a neighbourhood of training samples is taken into account in the parameters space, whereas nearest-neighbour simply returns the sample with highest score. Another obvious advantage is that the formulation of a regression-based method and its sparsity make it much faster than nearest neighbour classification.
7.4 Learning to Relate Descriptors to 3D Poses

To relate the image descriptors $x_i$ to the 3D joint angles and pose settings $y_i$, Agarwal and Triggs [AT04a] proposed the use of a regression method that learns the relation between $I$ pairs of vectors $(x_i, y_i)$ by estimating the coefficients or weights of a linear combination of basis functions $\phi_k$. The problem is described as:

$$y_i = \sum_{k=1}^{p} a_k \phi_k(x_i) + \epsilon = Af(x_i) + \epsilon$$  \hspace{1cm} (7.2)

where $\epsilon$ is a residual error vector, $y_i \in \mathbb{R}^m$ ($i = 1, 2, \ldots, I$), and $a_k \in \mathbb{R}^m$ ($k = 1, 2, \ldots, p$). For compactness, the weight vectors can be gathered into an $m \times p$ matrix $A \equiv (a_1 ~ a_2 ~ \cdots ~ a_p)$ and the basis functions into a $\mathbb{R}^p$-valued function $f(x) = (\phi_1(x) ~ \phi_2(x) ~ \cdots ~ \phi_p(x))^\top$. As discussed later, $p = K$ for linear kernel, and $p = I$ for Gaussian kernel. In order to estimate the bias of the samples in the state space, one can use $f(x) = (1 ~ \phi_1(x) ~ \phi_2(x) ~ \cdots ~ \phi_p(x))^\top$ and add a weight parameter to be estimated, but this is unnecessary if the data is standardized to have zero mean and unit standard deviation.

For $I$ training pairs, the estimation problem takes this form: estimate $A$ such that

$$A = \arg \min_A \left\{ \sum_{i=1}^{I} ||Af(x_i) - y_i||^2 + R(A) \right\} \hspace{1cm} (7.3)$$

where $R(\cdot)$ is a regulariser on $A$. Gathering the training vectors into an $m \times I$ matrix $Y \equiv (y_1 ~ y_2 ~ \cdots ~ y_I)$ and a $p \times I$ feature matrix $F \equiv (f(x_1) ~ f(x_2) ~ \cdots ~ f(x_I))$, equation (7.3) can be rewritten as:

$$A = \arg \min_A \left\{ ||AF - Y||^2 + R(A) \right\} \hspace{1cm} (7.4)$$

7.4.1 Regression with Relevance Vector Machines

For unidimensional signals $y$, Tipping [Tip01] proposed the use of Relevance Vector Machine (RVM), a method based on sparse Bayesian learning to estimate efficiently an $A_{(1 \times p)}$ with large sparsity. Each weight parameter is associated with an independent noise model $\alpha$ and there is a prior for $\alpha$ parameters (hyperpriors), which are modelled as Gamma functions, so they have a high probability near zero, enforcing sparsity in the estimate of the weights. Upon minimization, the regularization parameters push the weights $a$ of the less relevant basis functions to zero, thus producing a sparse model. This sparsity can save computational time and space.
A straightforward adaptation of this method for multidimensional state vectors $Y$ can be achieved by regressing input vectors $x$ against each of the $m$ individual elements $y_j$ of $y$ separately and concatenating the obtained row vectors of weights into matrix $A_{(m \times p)}$.

An experiment with the open-close data set was performed using $K = 90$ (i.e. $K = 30$ for each view) and linear kernel functions ($f(x) = x$). During the optimization, weight values $a$ that were smaller than a threshold $T_a$ where set to zero. $T_a$ was set to a value that give an average of 10 non-zero weights for each dimension $y_j$. The resulting non-zero elements of $A$ matrix are represented in Figure 7.16. The application of the obtained regressor on samples from the training set resulted on the mean absolute error\(^3\) of $3.1^\circ$, and mean standard deviation of $2.3^\circ$. The worst result was obtained with the interphalangeal\(^4\) joint of the thumb, which is occluded in many of the training images. For that joint angle, the average error was $9.5^\circ$ and the standard deviation was $6.9^\circ$.

A problem with regressing parameters independently is that noisy data potentially provide impossible output poses. For example, a regressor trained to recover 3D pose of walking humans might output poses having both legs to the front.

### 7.4.2 Agarwal and Triggs’ Regression Method

The pose of each DOF of the hand in natural motion without external forces is clearly not independent from the pose of the others. In [AT06b], Agarwal and Triggs describe an adaptation of Tipping’s method that estimate the whole matrix $A$ in a single process, creating a linear combination of relations with

\[\text{Computed by } \sum_i |A f(x_i) - y_i| / I\]

\[\text{For hand joints nomenclature, see Figure 1.1 and [Stu92].}\]
7.4 Learning to Relate Descriptors to 3D Poses

multi-dimensional output. This regressor is estimated by direct optimisation of the weights keeping the hyperprior parameters fixed.

The first step of this algorithm is to initialise $A$ with ridge regression. The regulariser is chosen to be $R(A) \equiv \lambda ||A||^2$, where $\lambda$ is a regularisation parameter. The problem can be described as the minimisation of

$$||A\bar{F} - \bar{Y}||^2 = ||AF - Y||^2 + \lambda ||A||^2,$$

(7.5)

where $\bar{F} \equiv (F \lambda I)$ and $\bar{Y} \equiv (Y 0)$. $A$ can be estimated by solving the linear system $A\bar{F} = \bar{Y}$ in least squares, i.e., $A \leftarrow [(FF^T)^{-1}F^TY]^T$. Ridge solutions are not equivariant under scaling of inputs, so both $x$ and $y$ vectors are scaled to have zero mean and unit variance before solving. The mean and standard deviation of the components of $x$ and $y$ are kept for application on testing data.

The next step is to apply a modification of Tipping’s RVM regression method. Instead of modelling $p(\alpha)$ (the hyperpriors of the weights $a$) as Gamma functions, Agarwal and Triggs use $p(a) \approx ||a||^{-\nu}$, which is a simple case of Gamma function with constant parameters. Their method is a maximum a priori type and directly optimises the weight parameters $a$, while Tipping’s is a type-II maximum likelihood approach that integrates out the parameters and optimizes the hyperparameters.

In order to reduce the risk of premature trapping of weight parameters to zero and overfitting, Agarwal and Triggs proposed to successively approximate the penalty terms with quadratic “bridges”. Therefore, with $a$ an element of $A$, the regularisers $R(a) = \nu \log ||a||$ are approximated by $\nu \left( ||a||/a_{scale} \right)^2 + const$, where $a_{scale}$ is a constant that is updated at each iteration\(^5\). The approximation has the same gradient as the original function at $a = a_{scale}$, and if $const$ is set to $\nu \left( \log ||a_{scale}|| - \frac{1}{2} \right)$, the same values at $a = a_{scale}$, as shown in Figure [A17].

Agarwal and Triggs proposed the use of column-wise set of priors in the regulariser $R(A)$: with a a column of $A$, $R(a) \approx \frac{\nu}{2} \left( ||a||/a_{scale} \right)^2 + const$, implying that the estimated matrix $A$ has some columns tending to zero as the method iterates, $a_k \approx 0$. Depending on the kernel function used, two different aspects of cost reduction for pose estimation can be achieved:

- If linear basis functions are used, i.e., $f(x) = x$, the zero vectors $a_k$ indicate which components

\(^5\)In [A T04a] and [A T04c], the authors have missed the division by two in the approximation of the regularizers $R(a)$, but this has been corrected in [A T04b] and [A T06b].
of vectors $x$ can be removed without compromising the regression result. Therefore, RVM can be used as a feature selection method, resulting in a reduction in the number of shape descriptors needed.

- Alternatively, kernel basis functions can be used. They are expressed by $\phi_i(x) = \mathcal{K}(x, x_i)$, making $f(x) = [\mathcal{K}(x, x_1), \mathcal{K}(x, x_2), \cdots, \mathcal{K}(x, x_n)]^\top$, where $\mathcal{K}(x, x_i)$ is a function that relates $x$ with the training sample $x_i$. For example (as used in this chapter), one can use a Gaussian kernel $\mathcal{K}(x, x_i) = e^{\beta ||x - x_i||^2}$, with $\beta$ estimated from the scatter matrix of the training data. In this case, the column-wise sparsity of $A$ acts as a method to select relevant training samples.

The estimation of $A$ is then performed in a similar fashion as to Equation 7.5, by iteratively solving the linear system:

$$A(FR) = (Y\ 0)$$

(7.6)

where $0$ is a $m \times p$ matrix of zeros and $R$ is a $p \times p$ matrix whose rows are defined by $\nu/a_{\text{scale}}$, and $a_{\text{scale}}$ is the norm $||a||$ of each column vector of $A$ from the previous iteration. To reinforce sparsity, the columns of $A$ whose norms are small are set to zero. This process is repeated until $A$ converges.

Figure 7.18 shows the $A$ matrix obtained by this method using linear kernel functions in the open-close data set, with multiple view 90-d descriptors $x$. The threshold $T_A$ on the norm of the column
Figure 7.18: Map of non-zero elements of matrix $A_{(m \times p)}$ resulting from linear regression using Agarwal and Triggs’ method, selecting 10 relevance vectors in total (a). The result obtained with RVM of individual parameters shown in Figure 7.16 is reproduced in (b) to show that some agreement between the two methods is obtained on the selection of columns of $A$. 
vectors $||\mathbf{a}||$ was tuned to select 10 relevant features, resulting in the selection of 5 features from camera 1 (side view), 3 features from camera 2 (top view), and 2 features from camera 3 (another side view). For samples in the training set, regression with this matrix resulted in a mean absolute error of $2.7^\circ$, and a mean standard deviation of $2.0^\circ$. The worst average error and standard deviation were $11.8^\circ$ and $8.2^\circ$ respectively, both for the interphalangeal joint of the thumb. This represents an improvement in comparison to the results obtained by regressing the DOFs individually. As discussed later, the column-wise sparsity of matrix $\mathbf{A}$, allows the application of this method for feature selection or sample selection. It is interesting to note that many of the vectors selected using Tipping’s method coincide with rows selected by Agarwal and Trigg’s method, confirming the common theoretical basis of both.

### 7.4.3 Applying the Regressor with Feature and Samples Selection

It has been observed that Gaussian kernel functions can provide better results at the expense of being slower than linear kernel functions [504a]. Indeed, the results showed later suggest that linear functions are less stable to noise than Gaussian kernel functions. The alternative proposed here is to combine both by first reducing the dimensionality of the image descriptors $\mathbf{x}$ with feature selection and then using regression with Gaussian kernel functions to select the most relevant samples. Since the dimension of the vectors $\mathbf{x}$ is reduced in the first stage, all the distance calculations required to compute $f(\mathbf{x})$ with Gaussian kernels is sped up.

After the training process has been performed to obtain matrix $\mathbf{A}$ and the sets of selected features and samples, the algorithm shown in Algorithm 3 is applied to estimate the hand pose given a new (set of multiple view) image(s). Note that, although the initial steps are not affected by feature selection and sample selection, these have a large impact on steps 6 to 13 (specially 11). Thus a trade-off between speed and robustness can be achieved.

### 7.5 Experiments and Results

This section presents experiments on applying regression for feature selection, samples selection and both combined. Aiming to perform a fair comparison, for both single-view and three-views data, 90-dimensional $\mathbf{x}$ descriptors were used, with the difference that, for the former, all the elements of $\mathbf{x}$ were obtained from the same view, and, for three-views, each view was described by a 30-dimensional vector,
Algorithm 3 Pose Estimation with Selected Features and Samples (application phase)

Require: lists of selected features, selected samples $x_i$, shape contexts centroids, means $\overline{x}$ and $\overline{y}$, $\text{std}(x)$, $\text{std}(y)$, and matrix $A$

1: for each camera $c$ do
2: if there are selected centroids from this view then
3: Extract the hand silhouette using skin detection and edge detection
4: Compute all the shape context vectors from all the silhouette contour points
5: Calculate their distances to all the centroids of this view
6: Soft histogramming: create the image descriptor $x'^c$ taking into account only bins related to the selected centroids
7: end if
8: end for
9: Concatenate the vectors $x'^c$ into a single vector $x'$
10: Standardise it using the mean $\overline{x}$ and $\text{std}(x)$ from the training set
11: Evaluate functions $\phi_i(x')$ to build $f(x')$, where $i$ is the index of kernel functions related only to the selected training samples
12: Apply $y \leftarrow Af(x')$ with the selected columns of $A$ only
13: ‘De-standardise’ $y$, using the mean $\overline{y}$ and $\text{std}(y)$ from the training set

and they were concatenated to build $x$. The parameters $\lambda$ and $\nu$ were both set to 0.3, which, in most of the experiments in this chapter, lead to convergence after three iterations for linear kernel functions, and after five iterations for Gaussian kernel functions.

### 7.5.1 Number of Relevance Vectors

The graphs of Figure 7.19 show the number of selected relevance vectors as a function of the threshold $T_a$. Note that the same threshold leads to the selection of more relevance vectors for a single view. This hints that even though the same number of training samples (and of the same dimensionality) is used in both cases, fewer relevance vectors are selected for multiple views, indicating that their measurements are more discriminative. Fewer samples and fewer features are needed to achieve the same relevance for multiple views. A similar behaviour has been observed for the complex training set.

### 7.5.2 Synthetic Images

For the experiments with synthetic images, ground truth is available. The data set was evenly split in a training set and a testing set (with no intersection) from the same sequence of movements. Although this practice makes training and testing data very similar, it is enough to distinguish the performance between single and multiple view methods.
Figure 7.19: Number of selected relevance vectors for linear and Gaussian kernels for single and multiple views as a function of the threshold $T_a$ evaluated for both the open-close and complex training sets.
7.5 Experiments and Results

Figure 7.20: Silhouettes obtained from a sample pose in the training set from camera 1 and 2, highlighting (with red ‘*’*) the points whose shape context is taken into account after the selection of two relevant features.

Open-Close Data Set

The sequence of movements in the open-close dataset can roughly be described by two degrees of freedom: flexion of the all joints and twisting movement of the hand about the forearm axis ($\theta_Z$). In order to verify the ability of the regressor to identify this, a feature selection experiment was performed, i.e., a regressor with linear kernel functions was trained with the threshold $T_{\alpha}$ on $||a||$ tuned to select only two relevance vectors. But for a single-view, three features were selected, because any greater threshold resulted on only one feature. For three-views, one vector from the top view and another vector from one of the side views (camera 1) were selected, as shown in Figure 7.20 and 7.21.

The points of the silhouette shown by red ‘*’ in Figure 7.20 are those whose shape contexts (SC) have the selected centroids among their four nearest centroids in SC space (the soft histograming implemented considers a neighbourhood of four centroids). Note that, for both views the centroids selected are close to the wrist rather than the finger tips. A possible reason for that is that features closer to the finger tips present too much variation between samples and they are not present for some of the samples, like those with the hand in fist pose. This was also observed for single view.

Figure 7.22 illustrates the result by showing the estimated angle of the interphalangeal joint of the index finger and the results for $\theta_Z$. Recall that at each 78 frames the images were generated for a different value of $\theta_Z$ (global hand orientation). Note that the pose of the hand was estimated individually for each frame, which explains the jittering motion.
30 clusters of all the shape contexts from camera 2

![Shape contexts manifold with the centroid of the selected cluster from camera 2 indicated by a blue circle.](image)

**Figure 7.21**: Shape contexts manifold with the centroid of the selected cluster from camera 2 indicated by a blue circle.

![Regression results using only three (for single view) and two (for multiple views) relevance vectors (out of 90), with a linear kernel and synthetic images: (a) estimated angle of the interphalangeal joint of the index finger; (b) estimated angle $\theta_Z$ of global rotation about the forearm axis.](image)

**Figure 7.22**: Regression results using only three (for single view) and two (for multiple views) relevance vectors (out of 90), with a linear kernel and synthetic images: (a) estimated angle of the interphalangeal joint of the index finger; (b) estimated angle $\theta_Z$ of global rotation about the forearm axis.
Figure 7.23: Regression results using only 10 samples (out of 273), with Gaussian kernel functions and synthetic images: (a) estimated angle of the interphalangeal joint of the index finger; (b) estimated angle $\theta_Z$ of global rotation about the forearm axis.

These results show that the regressor is able to give a rough approximation of the pose using a minimal set of selected vectors (in this case, image features). Even using fewer features for multiple views it is possible to achieve higher accuracy than with a single view. Also, the results for $\theta_Z$ with a single view seem to have no correlation with the ground truth. Furthermore, for single view, as $\theta_Z$ grows, the estimate for other angles gets poorer because the top view does not offer enough distinct features on its own when the fingers get nearly aligned to the camera axis.

When using Gaussian kernels, it is harder to intuit the minimal set of samples needed to estimate the pose. $T_a$ was chosen so that 10 relevant samples were selected from the training set. The trajectories obtained from single and multiple views are shown in Figure 7.23 in comparison with the ground truth.

Both for single and multiple views, the selected samples are mostly from “near-fist” hand poses. This may seem odd, but it is not usual in RVM for the the most relevant samples to be distant from the obtained pose estimates, and for them not to be the most comprehensive samples in terms of the variability of state (poses) [Tip01].

Figure 7.24 reports the application of feature selection followed by samples selection to combine speed and performance. Note that the superiority obtained for multiple views is more evident for $\theta_Z$. 
Complex Data Set and Quantitative Results

The complex data set incorporates a large range of hand poses, so it is more difficult to illustrate the results with graphs as shown for the open-close data set. Figure 7.25 shows the mean and standard deviation of the error for each parameter (DOF) of the hand for the complex data set, using half the samples for training and half for testing with both feature selection and samples selection. In this case, 36 relevance vectors with 35 dimensions were selected for both single and multiple view. For a comparison, the STD of the training trajectory is also shown. Note that both the greatest variation in angle and the greatest average error occur in the proximal interphalangeal joint of the fingers. In this database, the use of multiple views reduces the error in a roughly uniform manner along the pose parameters.

Table 7.1 shows a quantitative evaluation of the results for both data sets using synthetic images. The columns ‘ftrs.’ and ‘spls.’ indicate how many relevance vectors were selected with linear and Gaussian kernel, respectively. The column ‘worst result’ shows the average error for the parameter (DOF) whose estimate was the worst, indicated in the column ‘which DOF’. The abbreviation T IP refers to thumb’s inter-phalangeal joint, and M DIP to the distal inter-phalangeal joint of the middle finger.

As expected, the worst estimates occurred in two cases: (i) for DOFs related to parts of the hand whose contour was occluded in many of the images, and (ii) for the rotation $\theta_Z$ when a single view is
Figure 7.25: Panel (a) shows the standard deviation (STD) of the value of each hand parameter along the trajectory of the *complex* data set. Parameter 1-6 are absolute pose, 7 and 8 are wrist angles, 9 is abduction of the little finger and 10-12 are flexion angles. The same pattern repeats for each of the other fingers and thumb. Panel (b) shows the mean error and STD for each parameter using a single top view. Panel (c) shows the same for multiple views.
Table 7.1: Results with synthetic data obtained using 273 and 839 training samples for open-close and complex data sets, respectively. The same amount of samples was used for testing, though there is no intersection between the sets. For both data sets, the total number of features used is 90.

used, as this is not a rotation parallel to the top view image plane.

In general, the improvement obtained by using multiple views is evident, particularly when the number of features used is small. However the improvement is view-dependent, and if a single view captures the most meaningful silhouette the improvement is diminished. A further reduction in improvement arises because the synthetic images used so far are noise free. As shown in next section, improvement is restored when using real images.

7.5.3 Real Images

For real images, whole training sets were used, giving 546 training pairs for the open-close data set and 1679 for the complex data set. For testing, images of the right hand of a single subject were used. Since there is no ground truth available for the real images, only qualitative results are shown.

Figure 7.26 shows the estimated index PIP joint and $\theta_Z$ angles obtained by training the regressor with the open-close data set and applying it to the nine images shown in Figure 7.15. It is intuitive to visualise the correctness of these results, as both the estimated index PIP joint and $\theta_Z$ angles are expected to grow with time. In this case, the use of multiple views does not seem to show an improvement in relation to single view.

However, Figure 7.27 shows that multiple views provide a significant improvement for images with
more complex movements, using the regressor trained with the complex data set. This improvement becomes more evident when a small selection of features and samples is used, as shown in Figure 7.28. Note that, for a single camera, the regressor seems to be unable to recover some of the poses, probably because the measurements generate poses that extrapolate the space of trained poses.

### 7.5.4 Computational Cost

As the aim is to use this system for (re-)initialisation of a video-rate hand tracker, the computational cost is evaluated in this section. Although most components of this system were implemented using an interpreted language (MatLab 6.5, except where indicated), the time measurements presented here give a good clue of the computational complexity of each part of the algorithm. These experiments were performed using a computer with two 2.4GHz Pentium 4 CPUs and 750MB of RAM running Red Hat 9 Linux (though the algorithm was not parallelised).

#### Feature Extraction

This is the first step to obtain image descriptors, both for training and testing samples. For 5037 images of $120 \times 90$ pixels, the average time for skin colour detection was 2.8ms using a compiled C++ implementation. To extract subsampled silhouette contour points and calculate their shape contexts it takes...
Figure 7.27: Results obtained from real images (top row) for single view (middle row) and multiple views (bottom row), using Gaussian kernel with all the samples and all the features.

Figure 7.28: Results obtained from real images (top row) for single view (middle row) and multiple views (bottom row), using combined linear kernel to select 32 features (out of 90) and Gaussian kernel to select 38 samples (out of 1679).
7.5 Experiments and Results

![Table 7.2: Training time for the complex data set (1679 samples) for linear, Gaussian and the combined kernels to select 32 features and 38 samples.](image)

Table 7.2: Training time for the complex data set (1679 samples) for linear, Gaussian and the combined kernels to select 32 features and 38 samples.

Further 141.7ms per image in MatLab. Next, to calculate the quantised vectors $\mathbf{h}$, it takes 26.4ms per image. Therefore, the average time for this pre-processing stage is 170.9ms and this is the only stage where the use of multiple view can represent a linear ($O(C)$, where $C$ is the number of cameras) increase in the amount of time required by the algorithm.

**Training Phase**

The most demanding step of the training phase is the clustering to obtain the centres for vector quantisation. With the complex data set the algorithm did not converge until the maximum number of iterations (30) was reached. For this, $K$-means can take between one and ten days, depending on how many iterations of the second phase were performed, and this is data dependent [Seb84]. For the open-close data set, a result was obtained between 30 mins and 2 days, again, depending on the view and the number of second phase iterations performed. No convergence was reached within 30 iterations, but experiments have shown that the quality of the centroids for vector quantisation does not affect the discrimination power of the obtained vectors [JT05].

Given the training samples represented by the (concatenated) quantised vectors $\mathbf{x}$, to train Agarwal and Trigg’s regressor is a much faster process, as shown in table 7.2. Note an improvement in speed using the linear kernel functions followed by Gaussian kernel functions (both) in relation to using Gaussian kernel functions only.

**Application**

As shown in table 7.3, the actual pose estimation process is extremely fast (note that the scalars are in microseconds). For both for training and application, the difference between using both kernels and Gaussian kernel functions only is not very significative in comparison with the difference between these and the use of linear kernel functions only. However, combining both methods give the robustness of
Table 7.3: Average time over 1679 trials for the application of the regressor to the complex data set using linear, Gaussian and the combined kernels. Note that the time scale is in microseconds (µs).

<table>
<thead>
<tr>
<th>Kernel</th>
<th>Features</th>
<th>Training Samples</th>
<th>Time (µs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear</td>
<td>32</td>
<td>1679</td>
<td>7.2</td>
</tr>
<tr>
<td>Gaussian</td>
<td>90</td>
<td>38</td>
<td>35.7</td>
</tr>
<tr>
<td>Both</td>
<td>32</td>
<td>38</td>
<td>25.4</td>
</tr>
</tbody>
</table>

Gaussian kernel functions and the additional reduction on the risk of overfitting if all the features are used.

Summing up all the steps give, in the worst case (three cameras and Gaussian kernel functions) 651.69ms per frame, which is a very good result for a global detector with no prior information of the hand pose, implemented mostly in an interpreted language. Better results are expected in a compiled version, but the use of this method for detection only to (re-)initialise a generative tracker is still the best option for better results at lower computational cost.

**Memory Usage**

For application, this algorithm is not very memory demanding. Below is the list of required data:

- list of selected features: \( O(K) \), \( K \) is the dimension of \( x \);
- selected samples \( x_i \): \( O(K \times I) \), \( I \) is the number of training samples;
- shape context centroids: \( O(K) \);
- mean and standard deviation for vectors \( x \) and \( y \): \( O(K) \) and \( O(m) \), where \( m \) is the dimension of the state vectors \( y \);
- matrix \( A \): \( O(m \times K) \) for linear kernel functions or \( O(m \times I) \) for Gaussian kernel functions.

For a new image, in order to get \( x' \) (steps 4 to 6 of Algorithm 3), first it is necessary to obtain all the shape contexts from the silhouette contour. These are \( O(r \times a \times n) \), where \( r \) and \( a \) is the number of radial and angular bins of the shape contexts. In this chapter, \( r = 5 \) and \( a = 12 \); \( n \) is the number of points in the silhouette contour. For the complex data set the average number of points per image is 2549.1 per image in the experiments of this chapter, which is not a high value for today’s computers.
However, the training phase can be particularly demanding, specially in the clustering step, where it is convenient to keep all the shape context of all the training images in the memory: $O(r \times a \times n \times I)$. With the complex data set, 750MB of memory was enough for all the experiments (including training and clustering), in a MatLab implementation. But, for training, it was necessary process views individually and store data from the other views in the HD. Obviously, a significant amount of memory could be saved in a C or C++ implementation, in which numbers would not need to be represented with double precision for the calculations.

7.6 Conclusions

This chapter presented a regression-based method for estimation of hand pose in 3D from global image descriptors, advancing the single-view method of Agarwal and Triggs [AT04a] proposed for human pose estimation.

Skin silhouettes were extracted from colour imagery, and their contour points described using the shape contexts of Belongie et al. [BMP02]. The considerable variation in hand pose typically observed in imagery requires care to be taken to ensure scale and rotational invariance in the contexts. The use of contexts aligned with the axis of the forearm was found to be the best. By ensuring rotational and scale invariance, the number of training samples needed was reduced, provided triangulation was first used to recover the global pose parameters.

A global image descriptor for each view was obtained by coding the manifold of shape contexts using vector quantisation, and the descriptors combined at an intermiate level into multiview descriptors by concatenation. The mapping between multiview descriptors and 3D poses was learned using Agarwal and Triggs’ [AT04a] extension of Tipping’s Relevance Vector Machine [Tip01].

Our experiments have, inter alia, examined the effects of feature selection (linear kernel functions) and sample selection (Gaussian kernel functions) both on the quality of pose determination and on the computational time, using both synthetic and real imagery. We have found that linear kernel functions have the advantage of computational cost independent on the amount of training data used. However, we have found Gaussian kernel functions to be more robust, so we have performed experiments combining both linear and Gaussian kernels for speed and robustness. Our experiments have also shown that, for
7.6 Conclusions

general views, fewer relevance vectors are needed in the multiple view case. Their measurements are more discriminative, allowing correct pose estimates to be recovered in cases where a single view all but fails.

An obvious modification to the current image descriptor would be the use of a more sophisticated coding method, like Gaussian mixtures or Jurie and Triggs’s method \cite{JT05}. Another possibility is to explore the extension of RVM for multidimensional target spaces of Thayananthan et al. \cite{TNS+06} which, like the original RVM, optimises the hyperparameters. But the main thrust of future work should be to evaluate how relevant is the use of multiple hypotheses if multiple views are employed. A more application-oriented direction of this work is the integration with a generative tracker for real-time results, as done in \cite{AT05}, \cite{EZ05}, and \cite{RS06}. 