Nuisance Free Recognition of Hand Postures Over a Tabletop Display

João Carreira **Paulo Peixoto**

Institute of Systems and Robotics
University of Coimbra
Polo II - Pinhal de Marrocos
3030 Coimbra
Email: {joaoluis, peixoto}@isr.uc.pt

Abstract

This paper proposes a new approach to shape classification that is well suited to the specific challenges of vision-based hand posture recognition in a multi-user tabletop collaboration scenario. We use a representation of the 2-D hand silhouette where in-plane rotation and mirror symmetry appear as particular cases of permutations, and then show how to take advantage of this pattern to develop an efficient version of the permutation invariant SVM. Invariance to these transformations is very important because the users stand around the table, and a video camera captures the scene from the top. We also report experimental results that compare this approach favorably over common classification approaches, under the stated requirements.

Keywords: tabletop interaction, vision-based gesture recognition, support vector machines

1 Introduction

Tabletop displays have been a subject of considerable interest by the Human Computer Interaction community over the last fifteen years, as they present a natural medium for computer-assisted local collaboration between people. Computer Vision could be an important sensing technology for these systems, once it gets more stable: many users already have webcams which are cheap, easily deployable, and could be used to capture hand gestures at high frequencies. Also, LCD and Plasma displays are becoming larger and more economic, and cameras can adapt seamlessly to capture different screen areas. Tabletop systems present, however, very characteristic requirements to a gesture recognition software: full rotation invariance, because the users are around the table, and mirror symmetry invariance, to equally recognize left and right hand gestures. It should also be computationally cheap enough to cope with capturing multiple users' gestures simultaneously in real time and still allow the computer to run its applications. To understand these requirements consider the setups of the applications "Room Planner" (Wu & Balakrishnan 2003), and "CollabDraw" (Morris & Winograd 2006), illustrated in figure 1. These and other recent multi-user applications rely on newly developed multi touch sensitive tables, as Diamond Touch (Dietz & Leigh 2003), and use video projectors to provide the image. Although these sensors make a robust and dependable interface they're not without limitations, namely they're very expensive and can only directly capture information about the shape of the pressing areas of the hand against the table.

The focus of this paper is a mid-level vision problem: shape classification. We present a simple adaptation of a recent machine learning algorithm, the permutation invariant Support Vector Machine, or pi-SVM, (Shivaswamy & Jebara 2006), that combined with a properly coded representation of the silhouette of the hand, turns out to be an approach well suited to the specific problems of hand posture recognition in vision-based tabletop interfaces. In particular, it only distinguishes hand postures which differ intrinsically in shape, ignoring what we consider nuisance parameters: in-plane rotation and mirror symmetry. The approach doesn't require manual annotation of landmarks, although it requires labeled training data. The general idea of the pi-SVM is to, during training time, to transform the data in a way that both minimizes the radius of the hypersphere enclosing the points, and maximizes the margin between the points of the different classes, so that the nuisances get optimized away. Then, at test time, the transformation that best discriminates a pattern is first applied, followed by classification using a SVM. We show that in the desired tabletop setup, the nuisance parameters, to which the classification should be invariant, come up as restricted kinds of permutations, that can be handled by a less ambitious version of the pi-SVM. One that is also much more efficient than the original.

We assume there's some segmentation process that provides us with a closed contour of the hands. In our case we have been using skin color detection, which albeit improper for general situations, in the particular scenario where we're controlling the image in the LCD display it's a feasible solution.

The structure of this paper is as follows: we discuss related work in section 2, then the feature used is introduced in section 3, and customizations of the permutation invariant SVM are proposed in Section 4. Experimental results are shown, and discussed in section 5 and finally conclusions are presented in section 6.

2 Related Work

Most vision-based interfaces over a table have relied solely on fingertip tracking. For example, Letessier & Bérard (2004) matched a circular template to binarized images in order to detect fingertips independently of the orientation of the hand. Baraldi, Bimbo,

Research described in the paper was financially supported by FCT under grant No. POSC/EEA-SRI/61451/2004. The first author was financially supported by FCT PhD grant SFRH/BD/24295/2005.

(a) Room Planner

(b) CollabDraw

Figure 1: Examples of multi-user tabletop interaction setups.

Landucci & Valli (2006) used the same method, and built a simple rule-based classifier that can discriminate between three postures, based on the number of stretched fingers. Sato, Kobayashi & Koike (2000) use an infrared camera tuned to the human temperature to segment the forearm, and then uses the principal axe of the resulting blob to guide a normalized correlation search for the fingertips. This simplicity is a result of pragmatic thinking, as ambitious approaches from more traditional vision-based gesture recognition research don't work reliably and fast enough. A good review of these can be found in work by Derpanis (2004), which divides existing approaches as model-based, appearance-based and feature-based.

The types of invariances we seek have been mostly tackled using feature-based approaches, by pursuing representations of the features that directly incorporate them. A popular feature is the boundary of silhouettes, which has no internal holes or markings, making it easily representable in 1-D, parameterized by arc length. This kind of feature is incorporated in the MPEG-7 standard, and there's a large pool of solutions developed. There are representations of this feature that present some kinds of invariances, like Fourier Descriptors with respect to rotation. A more flexible feature is the Shape Context (Belongie, Malik & Puzicha 2002), which can represent a shape with inner markings, making it possible to use directly the output of edge extractors. The authors describe a way to achieve rotation invariance using this feature, but point out that it relies on contour tangents, which are highly sensitive to noise. Other flexible approach is the use of local invariant features that represent shape key points, as the SIFT feature (Lowe 2004), which can be directly calculated from the output of a low level interest point detector. The problem of this approach, as pointed out by Belongie *et al.* is that it sacrifices the shape information available in smooth portions of object contour, and that some objects - e.g. circles - don't even have any key points.

There's also work done on incorporating invariances directly into classification algorithms. Scholkopf & Smola (2002) identify three different approaches in the context of kernel methods: generating virtual support vectors, constructing invariance kernels and jittering support vectors. The first consists in generating virtual examples from the support vectors - informally speaking, the examples that are most difficult to classify - and then retraining using the new data. The virtual examples result from the application of transformations which we know *a priori* that shouldn't change the label of the example - the invariances. Invariance kernels work by directly regularizing the hyperplane in a way that trades off margin for parallelism to the directions of invariance. Finally, jittering support vectors works by transforming the example vectors in a way that the euclidean distances between them in feature space are minimal.

The problem with invariance kernels is that they can only be applied to smooth transformations. The virtual examples approach can become slow during classification time because of the increase in support vectors, and jittering can generate kernels that aren't positive definite, and so, the algorithms may not converge. A recently proposed method consists in cleaning up and reconstructing the data before training the classifier. This is the subject of the works by Bi & Zhang (2004) and Shivaswamy & Jebara (2006), with the latter resulting in the permutation invariant SVM. In order to make this paper more self-contained we introduce this algorithm in the next section, following the original presentation from Shivaswamy & Jebara (2006), while adding some additional comments from our analysis.

2.1 Permutation Invariant SVMs

The permutation invariant SVM is a binary classification algorithm, motivated by an important result in statistical learning theory (Vapnik 1995), which states that the expectation of the classification error probability is bounded by the ratio of the squared radius of the minimum hypersphere that encloses the data, to the square of the margin that separates the data points from both classes. This suggests a strategy to reduce the influence of unknown nuisance parameters in the data, by transforming the data along the desired invariances, and selecting the transformations that make the different classes most well separated, while being enclosed in a hypersphere with small radius.

In Shivaswamy & Jebara (2006) the targeted invariance was general permutation of the feature vector elements, and to this end the authors proposed calculating the radius and center of the minimal hypersphere enclosing the data, the maximal margin of the optimal separating hyperplane, and then, for each input sample, setting up a matrix of costs that indicates how favorable each different permutation is. The best permutation for each sample is then chosen by solving a Linear Assignment Problem (Papadimitriou 1982), or LAP, which can be done efficiently using the Kuhn-Munkres algorithm, also known as the Hungarian algorithm. After transforming all samples, the radius and center of the minimal hypersphere and the margin of the new optimal hyperplane are calculated again, and the rest of the process is repeated. After a number of iterations, the classifier corresponding to the optimal hyperplane of the transformed data is stored, in order to be used during test time. The training algorithm is described in Algorithm 1.

Computing the hyperplanes and hyperspheres can be done by solving the following optimization problems, present in most textbooks about kernel methods (Scholkopf & Smola 2002, Shawe-Taylor & Cristianini 2004). Let $\mathbf{w}$ be the vector of parameters of the hyperplane that separates two classes of data samples $\mathbf{x}_i$, with labels y_i, and ξ_i be slack variables for accounting noise, or non-separability of the classes.

Then the maximal margin hyperplane can be estimated from the solution of the following quadratically constrained quadratic program formulation:

$$\min_{\mathbf{w},b,\xi} \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{n}\xi_i \tag{1}$$

subject to

$$y_i\mathbf{w}'\mathbf{x}_i + b \geq 1 - \xi_i,\ \xi_i \geq 0\, \forall 1 \leq i \leq n \tag{2}$$

Similarly, the centroid and radius of the smallest hypersphere enclosing the data points can be estimated from:

$$\min_{\mathbf{c},R,\xi} R^2 + C\sum_{i=1}^{n}\xi_i \tag{3}$$

subject to

$$||\mathbf{c} - \mathbf{x}_i||^2 \leq R^2 + \xi_i,\ \xi_i \geq 0\, \forall 1 \leq i \leq n \tag{4}$$

In both cases the parameter C controls how acceptable it is for the margin and hypersphere radius to be violated, in order to account for noise. Large C corresponds to a hard margin and to the hypersphere containing all points.

For clarity's sake we believe it's useful to discuss the less obvious step of Algorithm 1: step 3. The idea is to find the permutation matrix that best transforms the feature vector, both in terms of how close it gets to the center of the hypersphere and to how far away it gets from the separating hyperplane. Consider the first term of the sum: if $\mathbf{w}$ is fixed, the dot product $\mathbf{w}'\mathbf{x}$ is proportional to the distance of $\mathbf{x}$ to the hyperplane, and that's what we want to maximize, by an appropriate choice of the permutation matrix A. If that was the only thing to optimize, the algorithm would proceed to find A by solving the maximization version of the LAP. Let $\mathbf{x}' = [x_1 x_2]$ and $\mathbf{w}' = [w_1 w_2]$. Then the reward matrix is:

$$\mathbf{wx}' = \begin{bmatrix} w_1x_1 & w_1x_2 \\ w_2x_1 & w_2x_2 \end{bmatrix}$$

The LAP, with this reward matrix, amounts to finding the one to one assignment of elements of $\mathbf{w}$ to elements of $\mathbf{x}$ such that the their sum is maximal. This $\mathbf{x}$ effectively maximizes $\mathbf{w}'\mathbf{x}$.

Conversely the aim of the second term is to minimize the dot product $\mathbf{c}'\mathbf{x}$. This corresponds to finding the permuted $\mathbf{x}$ which is closest to $\mathbf{c}$. For this to be true, $\mathbf{x}$ should be normalized to fixed length, and have positive elements (so that the minimal angle with $\mathbf{c}$ corresponds to a minimal distance, because $\mathbf{c}'\mathbf{x} = ||\mathbf{c}||\,||\mathbf{x}||cos\alpha$).

Finally, the λ parameter determines the trade off between optimizing the margin and the radius of the data enclosing hypersphere.

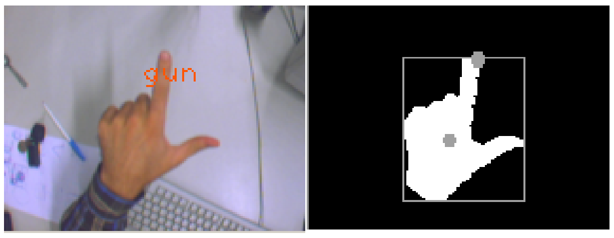
During test time, in order to predict the label of a test datum, the algorithm solves again the LAP problem for two different reward functions, $\lambda\mathbf{wx}' - \mathbf{cx}'$ and $-\lambda\mathbf{wx}' - \mathbf{cx}'$, getting in general two different permutations as solutions. The label corresponding to the largest absolute reward is then selected.

3 The Feature

The silhouette (S) of a segmented hand region, like the one depicted in 2, is a finite set of N_i points on the image, that define the basic shape of the hand (figure 3):

Algorithm 1 Algorithmic description of the original Permutation Invariant SVM

```
Input: Training data set - $(x_i, y_i)_{i=1}^n$, Maximum Iterations - $max$, Parameter - $\lambda$
Output: Hyperplane - $(w, b)$ and Centroid - $c$
0. Set $j \leftarrow 1$
1. Solve (3) from $(x_i, y_i)_{i=1}^n$ to find centroid $c^j$ and the radius $R$.
2. Solve (1) from $(x_i, y_i)_{i=1}^n$ to find $(w^j, b^j)$ and margin $M$.
3. Solve Kuhn-Munkres Algorithm with reward matrix $\lambda y_i w^j x_i' - c^j x_i'$ for each i, let the permutation matrix obtained be $A^{ij}$.
4. If $j = max$ return $(w^j, b^j, c^j)$ else $j \leftarrow j + 1$
```



(a) A posture. (b) Result of the segmentation.

Figure 2: Contour extraction using color segmentation.

$$S = \{s_k = (x_k, y_k), k = 1, \ldots, N_i\} \tag{5}$$

We assume that the silhouette S has the following properties:

- S is closed, i.e. s_1 is next to s_{N_i}.
- S has a depth of one single point (it's one dimensional).
- S is defined by accounting points in the clockwise direction.

The starting point of the definition of the representation, that we shall call a signature, is the calculation of the polar coordinates of each point s_k belonging to the contour of the segmented blob. The polar coordinates are defined in such a way that the origin of the coordinate system is the centroid $C = (c_x, c_y)^T$ of the segmented region R, defined as:

$$c_x = \frac{\sum_x\sum_y f(x,y)x}{\sum_x\sum_y f(x,y)}, \quad \text{and} \quad c_y = \frac{\sum_x\sum_y f(x,y)y}{\sum_x\sum_y f(x,y)}$$
$$\text{with } f(x,y) \begin{cases} 1 & \text{if } x,y \in R \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

Given the silhouette $S = (s_1, s_2, \ldots, s_{N_i})^T$ from the segmented hand on frame i we can compute the coordinates ρ_k, that corresponds to the Euclidean distance of each point to the centroid of the segmented hand blob, and θ_k, the angle:

$$\rho_k = ||s_k - C|| = \sqrt{(x_k - c_x)^2 + (y_k - c_y)^2} \tag{7}$$

$$\theta_k = \arctan\frac{(y_k - c_y)}{(x_k - c_x)}, \text{ with } k = 1..N_i \tag{8}$$

Having the polar coordinates, we split the silhouette S into r radial segments of equal size, and for

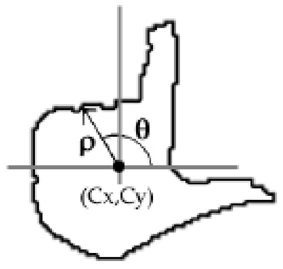


Figure 3: The silhouette's signature is defined by dividing the contour in fixed radial segments. The largest magnitude point in every segment is chosen. The center of the coordinate system is the centroid of the hand's blob.

each one we select the largest magnitude ρ'_k whose corresponding θ_k belongs to the angle interval that defines the segment. This way the signature has a fixed length of r elements. This signature is intrinsically invariant to translation of the hand in the image frame, since the silhouette is defined in relation to a coordinate system with its origin at the centroid of the hand's blob. The same is not true for scale: different distances from the camera to the hand will imply different silhouette amplitudes. A simple solution is very effective nonetheless: we normalize the ρ'_k coordinates in order to have them in the range $0 \le \rho'_k \le 1$. This is accomplished by dividing each ρ'_k by $\rho'_{max} = \max(\rho'_k)$, with $k = 1..r$.

In this way we get the final signature

$$signature(S) = [\frac{\rho'_1}{\rho'_{max}} \ldots \frac{\rho'_r}{\rho'_{max}}]' \tag{9}$$

This representation just lacks invariances to in-plane rotation and to mirror symmetry. Fortunately, these complex transformations in the image, translate to very simple transformations of the signature vector. Namely rotation in the plane perpendicular to the line that passes through the center of the camera sensor is mapped to a permutation P_r of the signature vector, up to orientation errors (due to the sampling from the silhouette) of $\frac{\pi}{r}$ radians. Mirror symmetry in that same plane is mapped to a permutation P_s.

In particular a rotation by an angle of $\frac{2\pi}{r}$ corresponds to the cyclic permutation:

$$P_r = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ \vdots & 0 & 1 & \ddots & \vdots \\ \vdots & \vdots & 0 & \ddots & 0 \\ 0 & \vdots & \vdots & \ddots & 1 \\ 1 & 0 & 0 & \cdots & 0 \end{bmatrix} \tag{10}$$

Rotations by $\frac{2n\pi}{r}$ map to P_r^n. For example, let $k = [k_1 \ldots k_m]'$ be a signature vector. Then $P_r^n k = [k_n \ldots k_m \, k_1 \ldots k_{n-1}]$.

The mirror symmetry across the line that passes through the centroid and the point whose magnitude is the first element of the signature vector is given by $P_s k$, with

$$P_s = \begin{bmatrix} 1 & 0 & \ldots & \ldots & 0 \\ 0 & \ldots & \ldots & 0 & 1 \\ \vdots & \ldots & 0 & 1 & 0 \\ \vdots & \iddots & \iddots & \iddots & \vdots \\ 0 & 1 & 0 & \ldots & 0 \end{bmatrix} \tag{11}$$

In this case $P_s k = [k_1 \, k_m \, k_{m-1} \ldots k_2]$.

4 The Classifier

As the kinds of invariance we desire are mapped to two specific permutations of the feature vector, our problem gets easier than the general Linear Assignment Problem. In fact, we go from $m!$ different possible assignments in the general permutation case, to just $m \cdot 2$, with m corresponding to the rotations, and the 2 to the mirror symmetry. The most efficient way to solve the LAP problem under this constraints is the evaluation of all hypothesis, which is $O(n)$, while the Kuhn-Munkres algorithm is $O(n^3)$.The only change required to algorithm 1 is then step 3: we should instead evaluate all the $m \cdot 2$ valid transformations of a signature, and choose the resulting permutation for which the reward $\lambda y_i w^j x'_i + c^j x'_i$, under an appropriate norm, is largest. The resulting situation can be better understood by seeing it as having a matrix of transportation prizes, from N factories to N warehouses, with the constraint that once you assign one factory to a warehouse, only two scenarios remain possible, and in both all the assignments are uniquely determined. For example, consider the following reward matrix:

$$R = \begin{bmatrix} 1 & 3 & 2 & 1 \\ 6 & 2 & 5 & 2 \\ 0 & 1 & 1 & 3 \\ 2 & 4 & 2 & 1 \end{bmatrix} \tag{12}$$

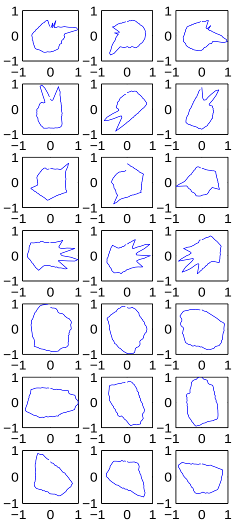
The LAP problem refers to finding the set of four r_{ij} elements, with no i and no j repeated, whose sum is maximal. This sum can be seen as an l1 norm. The solution in this case is $[r_{21} \, r_{42} \, r_{13} r_{34}]$ with total reward 15. Using our restrictions on the possible assignments, the solution would be $[r_{21} \, r_{12} \, r_{43} \, r_{34}]$ with total reward 14.

In order to illustrate the effect of the algorithm on the input signatures, and the meaning of different values of λ, it's useful to observe figure 4. In (a) 3 patterns from 7 different classes are initially with different rotations and mirror symmetries. In (b) are the same patterns after being transformed with a high λ, and in (c) with low λ. The SVM was trained in a one-vs-all scheme, with the one being the class of the patterns in the first row. The effect of high λ was to "encourage" a higher margin between the first class and all the others, and that is easily visible: the postures in the first class are oriented in a different direction than the others. In (c) they are all aligned, they're enclosed by a smaller hypersphere.

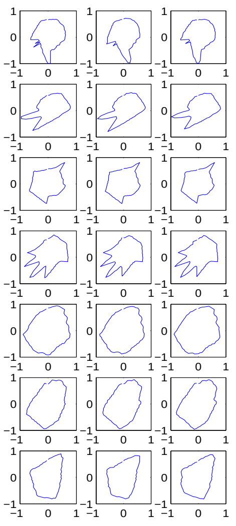
5 Experimental Results

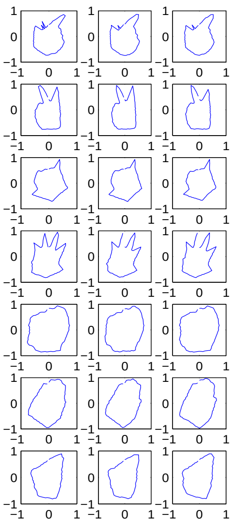
Due to the inexistence (to the best of our knowledge) of specialized image databases , we collected ourselves 50 samples from 7 different postures, with different scales, orientations and mirror symmetries. The samples represent hand gestures of 5 different adult male users, whose hand silhouettes were sampled to 80 points, after a skin color segmentation process. The postures considered are depicted in figure 5.

In order to evaluate the performance of the permutation invariant SVM on the data, we used it with a linear kernel, and compared against a normal SVM with a radial basis kernel applied on a regular feature vector, and on another feature vector which employed a popular heuristic to provide some invariance to rotation: selecting as the first value of the signature the one with largest magnitude and permuting cyclically the other elements of the signature accordingly. Using an SVM with this feature can, loosely speaking, be interpreted as an approximation to the permutation



(a)





(b) (c)

Figure 4: The effect of λ in the resulting patterns. In a) are the input patterns. In b) they are permuted after training a SVM for the class in the first row against the others, with high λ. In c) after training with low λ.

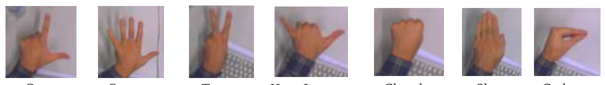


Figure 5: Predefined set of postures.

σ	SVM	SVM+heuristic	pi-SVM($\lambda = 0.001$)
0.005	62%	94%	95%
0.007	56%	83%	94%
0.01	46%	47%	70%

Table 1: Percentage of correct classifications with gaussian noise, with zero mean and standard deviation σ.

d	SVM	SVM+heuristic	pi-SVM($\lambda = 0.001$)
0.005	62%	86%	95%
0.01	56%	78%	88%
0.05	60%	68%	79%

Table 2: Percentage of correct classifications with salt & pepper noise (changes a d fraction of the signature points to magnitude 0 or 1).

invariant SVM with the restriction to permutations that rotate the feature, when solving the constrained LAP problem using the l_∞ norm. A C value of 2 was used in all experiments, so that we could focus on factors more directly connected to the algorithm under scrutiny.

We chose the number of iterations of the algorithm to be 4, because we observed that usually it was enough for convergence. The number of samples used in training was 10 from each class; the rest was used for testing. In order to to test the robustness of the different solutions we applied two kinds of noise, gaussian and salt and pepper (also known as on-off), and averaged the results over 5 sessions. In general terms, gaussian noise changes all the components of the feature vector by small amounts, while salt and pepper turns a few components to zero or one. The results are shown in tables 1 and 2, and the aspect of noisy examples is shown in figure 6.

Finally, we also tried different norms for solving the LAP problem - table 3. The measure of quality used in all experiments was the percentage of correct classifications in the test set.

5.1 Discussion of the Results

One curious observation was that for large λ the permutation invariant SVM performed very poorly. This was only verified in the multi-class scenario. Preliminary tests on two class discrimination didn't show this phenomenon, quite the opposite. This may be explained by a poor fit of the one-versus-all way of combining binary classifiers, which we employed.

For small λ the method performed better than the other solutions, specially when the data was noisy. In these cases it greatly outperformed the other methods.

We used a permutation invariant SVM with a linear kernel, and we think that using a kernel that creates a non-linear decision function can improve the performance of the method - especially if we can't

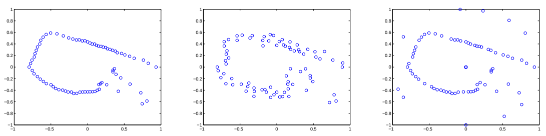


Figure 6: An example of the posture "two" is depicted on the left. In the center it's the same example, but with gaussian noise with mean 0 and standard deviation 0.005. On the right it's with salt and pepper noise, with 10 % of the components changed.

λ	pi-SVM(l_1)	pi-SVM(l_2)	pi-SVM(l_∞)
0.001	95%	99%	93%
0.1	90%	96%	88%
10	60%	74%	63%

Table 3: The effect of λ and the norm employed in solving the constrained LAP on the percentage of correct classifications.

solve the problem of transforming the data to have large margin - but this improvement comes with a performance price.

Of the norms employed in solving the constrained LAP problem, the one that behaved the best was the l_2 norm.

6 Conclusion

We presented a specialization of the permutation invariant SVM for classification of silhouette signature features. While the features used are too limited for general shape classification, because they are difficult to extract from images and cannot represent rich shapes - in particular those having important internal traits - they make a good fit for posture recognition over tabletops:

- They're cheap to compute, which is important in an input device (the mouse doesn't steal many cpu cycles).

- Certain interesting invariances appear as simple permutations of the feature vector, and this enables the use of the permutation invariant SVM efficiently.

- In tabletops powered by LCD displays, we can control the hand's background so that it is more easily segmentable.

The proposed method was shown to produce better results than other approaches, namely simple SVM classification with and without some common heuristics, using our data set of hand postures from five individuals. We have yet to evaluate the robustness of the approach to hand morphologies that are not in the database, like from children, but our preliminary results with synthetic noise looked quite promising.

Something that the proposed method apparently precludes is linear dimensionality reduction of the feature vectors (for example with PCA). The problem is that it's not possible to explore permutations of the features in a reduced linear space. We would have to transform the features back to the original space, perform the permutations and then transform the features back to the reduced dimensionality space. Maybe using nonlinear methods would work, like kernel PCA or spectral methods, but that would come with performance penalties.

Future work includes exploring different paradigms for combining the binary classifiers. One versus the rest doesn't appear to work well with the permutations of the data. A possibility is to experiment using the multi-class SVM(Weston 1999). We're also considering ways to extend the silhouette feature to include information about the internal traits of the shape.

References

Baraldi, S., Bimbo, A., Landucci, L. & Valli, A. (2006), wikitable: finger driven interaction for collaborative knowledge-building workspaces, *in* 'Computer Vision and Pattern Recognition Workshop, 2006 Conference on', pp. 144–144.

Belongie, S., Malik, J. & Puzicha, J. (2002), 'Shape matching and object recognition using shape contexts', *Pattern Analysis and Machine Intelligence, IEEE Transactions on* **24**(4), 509–522.

Bi, J. & Zhang, T. (2004), Support vector classification with input data uncertainty, *in* 'Advances in Neural Information Processing Systems'.

Derpanis, K. (2004), 'A review of vision-based hand gestures'.

Dietz, P. & Leigh, D. (2003), 'Diamondtouch: a multi-user touch technology', *ACM Symposium on User Interface Software and Technology (UIST)* **1-58113-438-X**, 219–226.

Letessier, J. & Bérard, F. (2004), Visual tracking of bare fingers for interactive surfaces, *in* 'UIST '04: Proceedings of the 17th annual ACM symposium on User interface software and technology', ACM Press, New York, NY, USA, pp. 119–122.

Lowe, D. G. (2004), 'Distinctive image features from scale-invariant keypoints', *Int. J. Comput. Vision* **60**(2), 91–110.

Morris, M.R., H. A. P. A. & Winograd, T. (2006), Cooperative gestures: Multi-user gestural interactions for co-located groupware, *in* 'Proceedings of CHI', pp. 1201–1210.

Papadimitriou, C. H. & Steiglitz, K. (1982), *Combinatorial Optimization: Algorithms and Complexity*, Prentice-Hall.

Sato, Y., Kobayashi, Y. & Koike, H. (2000), Fast tracking of hands and fingertips in infrared images for augmented desk interface.

Scholkopf, B. & Smola, A. J. (2002), *Learning with kernels: Support vector machines, regularization, optimization, and beyond.*, MIT Press.

Shawe-Taylor, J. & Cristianini, N. (2004), *Kernel Methods for Pattern Analysis*, Cambridge University Press.

Shivaswamy, P. & Jebara, T. (2006), Permutation invariant svms, *in* 'International Conference on Machine Learning'.

Vapnik, V. (1995), *The nature of statistical learning theory*, Springer-Verlag.

Weston, J., W. C. (1999), Support vector machines for multi-class pattern recognition, *in* 'ESANN'.

Wu, M. & Balakrishnan, R. (2003), 'Multi-finger and whole hand gestural interaction techniques for multi-user tabletop displays', pp. 193–202.