Gesture Recognition Based on Manifold Learning

Heeyoul Choi, Brandon Paulson, and Tracy Hammond

Dept. of Computer Science, Texas A&M University
3112 TAMU, College Station, TX 77843-3112, USA
{hchoi,bpaulson,hammond}@cs.tamu.edu

Abstract. Current feature-based gesture recognition systems use human-chosen features to perform recognition. Effective features for classification can also be automatically learned and chosen by the computer. In other recognition domains, such as face recognition, manifold learning methods have been found to be good nonlinear feature extractors. Few manifold learning algorithms, however, have been applied to gesture recognition. Current manifold learning techniques focus only on spatial information, making them undesirable for use in the domain of gesture recognition where stroke timing data can provide helpful insight into the recognition of hand-drawn symbols. In this paper, we develop a new algorithm for multi-stroke gesture recognition, which integrates timing data into a manifold learning algorithm based on a kernel Isomap. Experimental results show it to perform better than traditional human-chosen feature-based systems.

Keywords: Sketch Recognition, Manifold Learning, Kernel Isomap.

1 Introduction

Sketched gestures are a natural form of input for many domains, such as drawing mathematical symbols, graphs, binary trees, finite state machines, electrical circuit diagrams, military course of action diagrams and many more. To allow the computer to understand these diagrams, gesture recognition systems have been built for a large number of domain [1,2,3,4,5,6,7,8,9,10,11]. Many techniques for gesture recognition have been developed; however, current gesture recognition systems still struggle with trying to get high recognition accuracy while still providing drawing freedom. As in other recognition domains (such as image and speech), feature selection is crucial for efficient and qualified performance in gesture recognition. Previous research has provided several suggestions for good feature sets [12,13,14,11]. Rubine suggested 13 features based on stylistic drawing features and time [12]. Long modified Rubine's features by adding 11 new features and removing 2 time-based features [13]. Also, ink features were proposed in [11]. These feature sets were well designed, but they are based on manual entry of methods to extract them, which becomes tiresome.

As Mahoney says in [15], designing features manually is a tedious task and may be extended by designer's intuition, which is subjective. Moreover, feature design

N. da Vitora Lobo et al. (Eds.): SSPR&SPR 2008, LNCS 5342, pp. 247–256, 2008.

is sensitive to problems such as jitters. More interestingly, machine learning research in other domains have shown that computers are able to select their own features with similar results. As stated in [15], some researchers have tried to learn some features automatically as well as manually, which leads to a mixture of hand-chosen and machine-generated features. Human-generated features are not easily extendable, whereas it is trivial to add some dimensions in feature space for a computer. In addition, hand-chosen features are extracted based on human-observable properties, whereas machine-generated features can be optimized for clear advantages in classification.

In machine learning, the Hidden Markov Models (HMMs) [16], which use temporal and spatial structure, have been widely used for gesture recognition. Several sketch recognition systems have been built which use HMMs to help predict stroke ordering [17,18,19]. Bayesian Networks have also been applied [20] to recognize multi-stroke shapes using LADDER shape descriptions [21]. Some dissimilarity-based approaches have been proposed in [22] with image-based methods and in [23] with graph-based methods.

To our knowledge, however, manifold learning has not yet been applied to gesture recognition despite evidence that has shown manifold learning methods to be good nonlinear feature extractors. Manifold learning is an effective method for representation that works by recovering meaningful low dimensional structures hidden in high-dimensional data [24,25,26,27,28]. Previous research has attempted to use manifold learning to analyze 3D hand-gestures [29], but manifold learning has not yet been applied to gesture (or sketch) recognition.

In this work we apply the kernel Isomap manifold learning method to classify sketch data, because it has a projection property, which provides the ability to project (map) new test data into (onto) the same feature space as training data. Kernel Isomap requires a dissimilarity matrix to find a low-dimensional mapping from which it produces a feature set. We introduce a new algorithm to measure dissimilarity distance between shapes that is based on both spatial and temporal information. We also show how this algorithm can be modified to accommodate for shapes drawn with multiple strokes. For recognizing shapes, we compare our method with the widely used Rubine method [12] and the $1 recognizer [30], which is a recently proposed automatic sketch recognizer.

2 Kernel Isomap

Manifold learning involves inducing a smooth, nonlinear, low-dimensional manifold from a set of data points. Recently, various mapping methods (for example see [24,25,26,27]) have been developed in the machine learning community, and their wide applications have started to draw attention in pattern recognition and signal processing. Isomap is one of the representative isometric mapping methods, which extends metric multidimensional scaling (MDS), by using Dijkstra's geodesic distances (shortest paths) on a weighted graph, instead of Euclidean distances [25].

The geodesic distance matrix used in Isomap can be interpreted as a kernel matrix. However, the kernel matrix based on the doubly centered geodesic distance matrix is not always positive semi-definite. *Kernel Isomap* [31] exploits a constant-shifting method such that the geodesic distance-based kernel matrix is guaranteed to be positive semi-definite as a *Mercer kernel* [32] matrix. This kernel Isomap has a generalization property, enabling us to project test data points onto an associated low-dimensional manifold, using a kernel trick as in kernel PCA [33], whereas, in general, most embedding methods (including Isomap, LLE, and Laplacian eigenmap) do not have such a property. See [31] for the details.

3 Implementation

3.1 Dissimilarity from Sketch Data

To apply the kernel Isomap to sketch classification, what we need is not the data points, but a dissimilarity matrix, which is enough to find a low dimensional space. The first step in our algorithm is to scale each character to be the same width and height. Then, we need a consistent (and large) number of points in each sketch. Thus, we interpolate points between any two consecutive points that are adjacent in time, but far away in distance to ensure that each gesture has the same number of points.

Dissimilarity. To calculate the dissimilarity matrix, we calculate the sum of squared distance between two points of the same order in each sketch as in Fig. 1. This is the squared dissimilarity in our algorithm. The dissimilarity between the ith sketch data and the jth sketch data, D_{ij}, is then given by

$$D_{ij} = \sqrt{\sum_{k=1}^{S} (d_k)^2}, \tag{1}$$

where d_k is the Euclidean distance between the kth points of two sketches, and S is the number of points in one stroke.

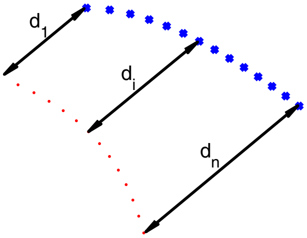


Fig. 1. Dissimilarity between two sketches

In addition to normalizing each stroke by resampling, we use different weights for different points. When we connect the two parts of '4' or '5', the connection causes confusion in calculating the distance. If we use different weights so that the fake points have less importance than real parts of the stroke, we can eliminate some of the confusion. Finally, the distance between two points is given by

$$D_{ij} = \sqrt{\sum_{k=1}^{S} (d_k)^2 w_{ik} w_{jk}}, \tag{2}$$

where w_{ik} is the normalized weight of the kth point in the ith stroke, which is given by

$$w_{ik} = \begin{cases} \frac{1}{Z_i} & \text{if the point is original in the stroke,} \\ \frac{\varepsilon}{Z_i} & \text{otherwise,} \end{cases} \tag{3}$$

where Z_i is the normalization term and ε is a constant between 0 and 1.

We call a kernel Isomap equipped with weighted distance *weighted kernel Isomap*[1]. The weighted kernel Isomap works well with multi-strokes shapes. For example, some characters such as the digit '7' might be drawn as one stroke or two strokes even by one user. In this case, weighted kernel Isomap works well since the weights make the boundary between one stroke and multi-strokes smooth so that the distance between two same characters can be closer.

There have been proposed some dissimilarities for sketch data [22]. However, those dissimilarities are based on images, while our proposed dissimilarity is based on sketch data including time information.

3.2 Classification

After getting features through kernel Isomap, we use the k nearest neighbor (kNN) method, which is a simple classifier, to check how good the features are[2] In kNN, we use Mahalanobis distance instead of Euclidean distance between points, since the variances of Rubine features are too different from each other, thus Euclidean distance does not find any meaningful structure from Rubine's features. On the other hand, in the case of kernel Isomap, the variances of the features are similar to each other, so Euclidean distance also shows good results.

4 Experiments

We carried out numerical experiments with three different data sets: (a) 26 small letters in English written by one user, where all characters are drawn by

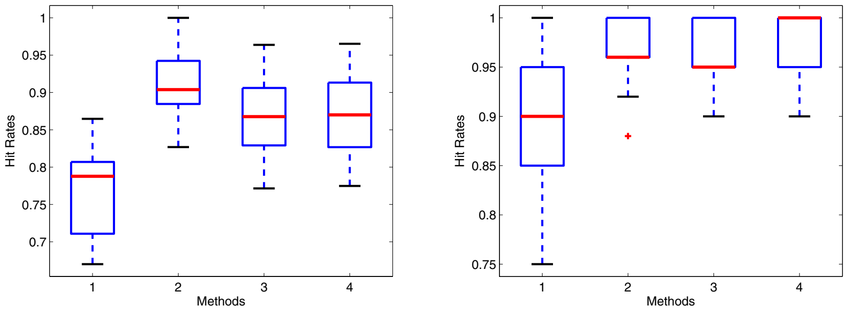
[1] When weighted kernel Isomap and kernel Isomap work in the same way, we name (weighted) kernel Isomap

[2] Applying kNN to the projected space does not give any significant benefit in classification compared to applying it directly to the input data space. However, the results from kNN is enough to show how much our methods have the information for classification after the projection to the feature space.

one stroke and each letter has 25 data points; a third of the examples of each letter were drawn two weeks later, so there is a reasonable amount of variability in the example data, (b) 10 digits from 0 to 9, drawn by another user, where each class has 25 characters and (c) 8 different mathematical symbols (‘+’, ‘-’, ‘x’, ‘/’, ‘=’, ‘sin’, ‘cos’ and ‘tan’), drawn by 10 subjects, where each class of each person has 5 characters, each of which is by one stroke. In order to show the improved accuracy of our method, we compared our method with Rubine’s algorithm and the \$1 recognizer.

4.1 Accuracies

For more robust results, we executed 10-fold cross validation 50 times. Figs. 2 and 3 show the hit rates of classification in three data sets, respectively, by three approaches: (1) Rubine’s method, (2) \$1 recognizer (3) kernel Isomap and (4) weighted kernel Isomap. To make it fair, we extracted 13 features from the (weighted) kernel Isomap as Rubine’s[3].



(a)

(b)

Fig. 2. For (a) alphabetical characters (b) digits, the classification accuracies of three methods: (Method 1) Rubine method (Method 2) \$1 recognizer (Method 3) kernel Isomap and (Method 4) weighted kernel Isomap

Fig. 2 is the boxplots of 50 time experiments in the cases of the English letters set and digits. In these figures, weighted kernel Isomap and kernel Isomap features are better than Rubine features in the classification, but similar to each other. Moreover, we can easily extend the number of features in (weighted) kernel Isomap by taking more eigenvalues and eigenvectors of the kernel matrix, which might produce better performance, whereas the number of Rubine features is 13 and it is hard to add another feature to them. Note that in (b), some digits such as ‘4’ and ‘5’ are composed of 2 strokes. This verifies that the weights for points in strokes work for the classification especially when characters are composed of multiple strokes. The average hit rates for (a) are 77.31%, 90.96%, 87.19% and 87.04%, respectively, and for (b) they are 91.80%, 97.28%, 95.90% and 97.80%, respectively.

[3] \$1 recognizer does not extract any features. Actually it is like using all features.

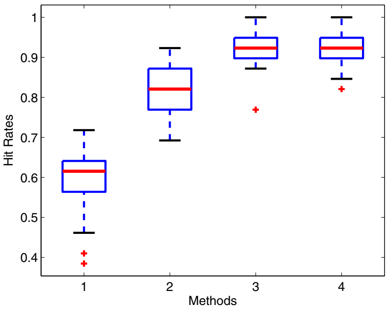


Fig. 3. For mathematical symbols (‘+’, ‘-’, ‘x’, ‘/’, ‘=’, ‘sin’, ‘cos’ and ‘tan’), the classification accuracies of three methods: (Method 1) Rubine method (Method 2) $1 recognizer (Method 3) kernel Isomap and (Method 4) weighted kernel Isomap. The average hit rates are 60.46%, 82.41%, 92.77% and 92.41%, respectively.

The previous two experiments used one user’s data set. As for the mathematical symbols, the data set was made by 10 different subjects. However, since each character in the data set was drawn by one stroke, there is no big difference between kernel Isomap and weighted kernel Isomap as in the first experiment. In Fig. 3, the accuracies of our proposed methods are much better than Rubine’s.

4.2 Feature Spaces

Figs. 4 and 5 show us how good the features are. We plotted the 4 best features from the English letters data set. In Fig. 5 which is from kernel Isomap, we can see that sketches in the same classes share similar feature values for most features, whereas in Fig. 4, which uses the Rubine feature set, just a few features are useful for classification.

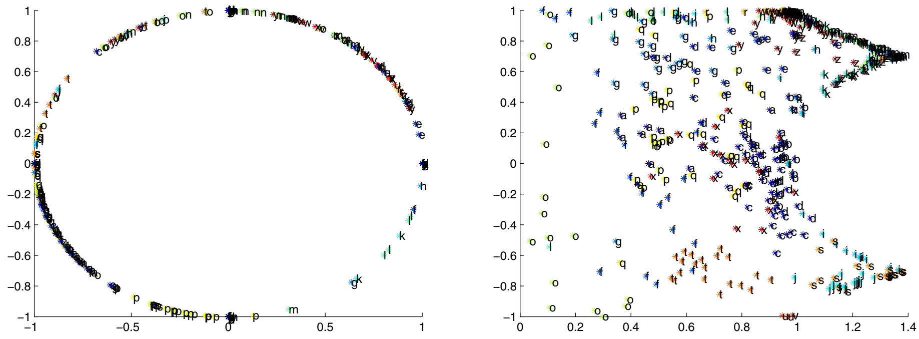


Fig. 4. Four features from Rubine’s. Each axis represents one Rubine feature. Note that the features do not effectively separate the different gestures.

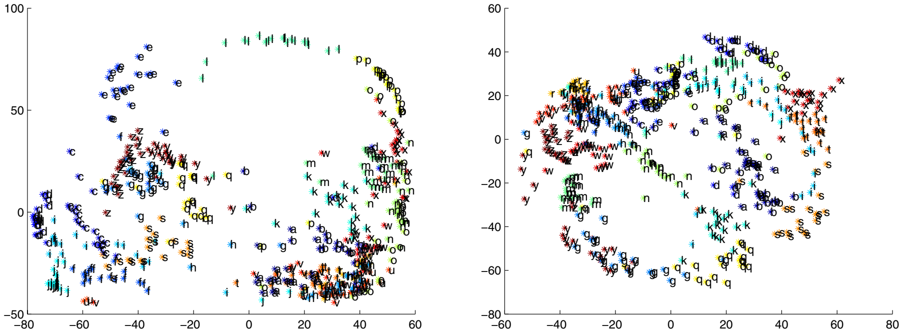


Fig. 5. Four features corresponding to the 4 largest eigenvalues of kernel matrix. Each axis represents an automatically generated feature. Note that the features effectively separate the different gestures.

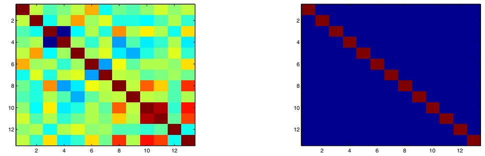


Fig. 6. Correlation matrices between 13 features (Red color represents high correlation and each axis represents 13 features): (Left) Rubine Features; (Right) (weighted) kernel Isomap Features. Note that in the figure on the right all of the automatically chosen Isomap features are highly uncorrelated, meaning that they are much more effective to include distinguishable information.

We calculated the correlation matrix of each feature set (see Fig. 6). (Weighted) kernel Isomap features are uncorrelated, which means they are more efficient than the heavily correlated and redundant Rubine features (seen particularly between features 10 and 11).

5 Discussion

As shown in the figures above, the features from (weighted) kernel Isomap are more accurate and more efficient than Rubine's for the classification. But, there are two drawbacks in the (weighted) kernel Isomap approach. To get the features, a kernel Isomap requires a lot of training data because it is assumed in manifold learning that data points should be dense. In addition, it takes some time to extract those features for training. However, after training, time does not matter when testing new data points. Also, although the Rubine's approach does not need dense data points for training, it also needs training data for the classification. More importantly, as we gather more data points, the features of a (weighted) kernel Isomap become better, whereas Rubine's features do not.

The weighted kernel Isomap works well with multi-strokes shapes. Sometimes, it is hard to say if the input character is composed of one stroke or multiple strokes. In this case, weighted kernel Isomap works well since the weights make the boundary between single stroke and multi-strokes smooth. There is some room to improve the classification hit rate by providing better weights.

The reason why (weighted) kernel Isomap is better than Rubine's for the classification is that they try to find the best features to represent the data set in an efficient way, whereas Rubine's features are already determined according to stylistic drawing properties. Therefore, the feature set of kernel Isomap reflects the domain or context dependent features.

With the letter data set which has large amount of orientation variability, the $1 recognizer is slightly better than our method because of their rotation invariant distance metric[4]. However, our method outperforms the $1 recognizer with math data set because it contained much less rotational variability. Furthermore, some shapes, such as '+' and 'x', contain rotational ambiguity which makes them hard to distinguish with $1. When a domain calls for rotation invariant features, we can improve our technique by simply using the distance measure in the $1 recognizer. Another disadvantage of the $1 recognizer is that it does not extract any features, so it can not improve anymore (it is fixed in its method of classification), whereas our method can be integrated with other types of classifiers, such as a quadratic classifier or support vector machine, which may improve overall performance. For the purposes of this work, we used a simple kNN classifier as proof that our manifold learning technique can be used to extract features from sketched data.

6 Conclusion

Gesture recognition is a natural form of human-computer interaction. As in other recognition problems, features are crucial for efficient and qualified performance in gesture recognition. In general pattern recognition problems, machines can learn effective features which may perform better than the hand-chosen ones in classification. Though, in many research areas, manifold learning methods have been shown to be good nonlinear feature extractors, only a few of them have been applied to gesture recognition. In this paper, we developed a new technique for gesture classification using a manifold learning approach that combines temporal and spatial information, while handling multiple strokes with weighted distances. Experimental results confirmed the performance to be better than the Rubine's feature set.

Acknowledgements

One of the authors was supported by StarVision Technologies's student sponsorship program and this work funded in part by NSF IIS grant 0744150: Developing Perception-based Geometric Primitive-shape and Constraint Recognizers to Empower Instructors to Build Sketch Systems in the Classroom.

[4] It is computationally so expensive to calculate rotation-invariant distances. So it takes much time even to classify a new point.

References

1. LaViola, J., Zeleznik, R.: Mathpad2: A system for the creation and exploration of mathematical sketches. ACM Transactions on Graphics (Proceedings of SIGGRAPH) 23(3) (2004)
2. Stahovich, T., Davis, R., Shrobe, H.: Qualitative rigid body mechanics. Artificial Intelligence (2000)
3. Landay, J.A., Myers, B.A.: Sketching interfaces: Toward more human interface design. IEEE Computer 34(3), 56–64 (2001)
4. Forbus, K.D., Usher, J., Chapman, V.: Sketching for military course of action diagrams. In: Proceedings of IUI 2003 (2003)
5. Do, E.Y.L.: VR sketchpad - create instant 3D worlds by sketching on a transparent window. In: de Vries, B., van Leeuwen, J.P., Achten, H.H. (eds.) CAAD Futures 2001, pp. 161–172 (July 2001)
6. Forsberg, A.S., Dieterich, M.K., Zeleznik, R.C.: The music notepad. In: Proceedings of UIST 1998, ACM SIGGRAPH (1998)
7. Igarashi, T., Matsuoka, S., Tanaka, H.: Teddy: A sketching interface for 3d freeform design. In: SIGGRAPH 1999, pp. 409–416 (August 1999)
8. Mahoney, J.V., Fromherz, M.P.J.: Interpreting sloppy stick figures by graph rectification and constraint-based matching. In: Fourth IAPR Int. Workshop on Graphics Recognition, Kingston, Ontario, Canada (2001)
9. Muzumdar, M.: ICEMENDR: Intelligent capture environment for mechanical engineering drawing. Master's thesis, Massachusetts Institute of Technology (1999)
10. Hammond, T., Davis, R.: Tahuti: A geometrical sketch recognition system for UML class diagrams. In: AAAI Spring Symposium on Sketch Understanding, March 25-27, pp. 59–68 (2002)
11. Patel, R., Plimmer, B., Grundy, J., Ihaka, R.: Ink features for diagram recognition. In: Sketch Based Interfaces and Modeling IEEE, Eurographics (2007)
12. Rubine, D.: Specifying gestures by example. Computer Graphics 25(4), 329–337 (1991)
13. Long, A.C., Landay, J.A., Rowe, L.A., Michiels, J.: Visual similarity of pen gestures. In: Human Factors in Computing Systems (2000)
14. Sezgin, T.M., Stahovich, T., Davis, R.: Sketch based interfaces: Early processing for sketch understanding. In: Proceedings of 2001 Perceptive User Interfaces Workshop (PUI 2001) (2001)
15. Mahoney, J.V., Fromherz, M.P.J.: Three main concerns in sketch recognition and an approach to addressing them. In: AAAI Spring Symposium on Sketch Understanding, Standord, CA, pp. 105–112 (March 2002)
16. Rabiner, L.R., Juang, B.H.: An introduction to hidden Markov models. IEEE Trans. Acoustics, Speech, and Signal Processing Magazine 3, 4–16 (1986)
17. Sezgin, T.M.: Sketch Interpretation Using Multiscale Stochastic Models of Temporal Patterns. PhD thesis, Massachusetts Institute of Technology (May 2006)
18. Sun, Z., Jiang, W., Sun, J.: Adaptive online multi-stroke sketch recognition based on hidden markov model. In: Yeung, D.S., Liu, Z.-Q., Wang, X.-Z., Yan, H. (eds.) ICMLC 2005. LNCS, vol. 3930, pp. 948–957. Springer, Heidelberg (2006)
19. Muller, S., Eickeler, S., Rigoll, G.: Image database retrieval of rotated objects by user sketch. In: IEEE Workshop on Content-Based Access of Image and Video Libraries, p. 40 (1998)
20. Alvarado, C., Davis, R.: Sketchread: A multi-domain sketch recognition engine. In: Proceedings of UIST 2004, pp. 23–32 (2004)

21. Hammond, T., Davis, R.: Ladder, a sketching language for user interface developers. Elsevier, Computers and Graphics 28, 518–532 (2005)
22. Kara, L.B., Stahovich, T.F.: An image-based trainable symbol recognizer for sketch-based interfaces. In: Making Pen-Based Interaction Intelligent and Natural, Menlo Park, California, October 21-24. AAAI Fall Symposium, pp. 99–105 (2004)
23. Lee, W., Kara, L.B., Stahovich, T.F.: An efficient graph-based recognizer for hand-drawn symbols. Computers & Graphics 31, 554–567 (2007)
24. Seung, H.S., Lee, D.D.: The manifold ways of perception. Science 290, 2268–2269 (2000)
25. Tenenbaum, J.B., de Silva, V., Langford, J.C.: A global geometric framework for nonlinear dimensionality reduction. Science 290, 2319–2323 (2000)
26. Saul, L., Roweis, S.T.: Think globally, fit locally: Unsupervised learning of low dimensional manifolds. Journal of Machine Learning Research 4, 119–155 (2003)
27. Belkin, M., Niyogi, P.: Laplacian eigenmaps for dimensionality reduction and data representation. Neural Computation 15, 1373–1396 (2003)
28. de Silva, V., Tenenbaum, J.B.: Global versus local methods in nonlinear dimensionality reduction. In: Advances in Neural Information Processing Systems, vol. 15, pp. 705–712. MIT Press, Cambridge (2003)
29. Jenkins, O.C., Matari, M.J.: A spatio-temporal extension to isomap nonlinear dimension reduction. In: Proc. Int'l. Conf. Machine Learning, Banff, Canada (2004)
30. Wobbrock, J., Wilson, A., Li, Y.: Gestures without libraries, toolkits, or training: A $1 recognizer for user interface prototypes. In: Proc. of the 20th Annual ACM Symposium on User Interface Software and Technology, Newport, RI, USA (2007)
31. Choi, H., Choi, S.: Robust Kernel Isomap. Pattern Recognition 40(3), 853–862 (2007)
32. Girolaini, M.: Mercer kernel-based clustering in feature space. IEEE Transactions on Neural Networks 13(3), 780–784 (2002)
33. Schölkopf, B., Smola, A.J., Müller, K.R.: Nonlinear component analysis as a kernel eigenvalue problem. Neural Computation 10(5), 1299–1319 (1998)