2014 IEEE International Conference on Systems, Man, and Cybernetics
October 5-8, 2014, San Diego, CA, USA

3D Hand Posture Recognition from Small Unlabeled Point Sets

Andrew Gardner,* Christian A. Duncan,† Jinko Kanno,* and Rastko Selmic*
*College of Engineering and Science
Center for Secure Cyberspace
Louisiana Tech University, Ruston, Louisiana 71272
Email: {abg010, jkanno, rselmic}@latech.edu
†Department of Mathematics and Computer Science
Quinnipiac University, Hamden, Connecticut 06518
Email: christian.duncan@quinnipiac.edu

***Abstract*—This paper is concerned with the evaluation and comparison of several methods for the classification and recognition of static hand postures from small unlabeled point sets corresponding to physical landmarks, e.g. reflective marker positions in a motion capture environment. We compare various classification algorithms based upon multiple interpretations and feature transformations of the point sets, including those based upon aggregate features (e.g. mean) and a pseudo-rasterization of the space. We find aggregate feature classifiers to be balanced across multiple users but relatively limited in maximum achievable accuracy. Certain classifiers based upon the pseudo-rasterization performed best among tested classification algorithms. The inherent difficulty in classifying certain users leads us to conclude that online learning may be necessary for the recognition of natural gestures.**

I. Introduction

In this paper we explore several classification algorithms for identifying static hand postures using position data provided by a Vicon motion capture camera system. The problem is complicated by the fact that positions (3D points) are provided as an unordered set, and due to occlusions and occasional extraneous markers, the size of the provided set of positions varies from frame to frame. One does not know *a priori* which physical features are represented among the set of positions. A natural complication arises when choosing features with which to perform classification, and choosing an appropriate feature extraction or transformation is challenging. Note that there is no context provided for the points other than each other. We will refer to the positions as raw features. Our comparison will include methods that work directly with the raw features as well as those that do not.

Our problem is distinct from that of image or point set registration [1], which aims to align several images or point sets via rigid or non-rigid [2] transformations. For all intents and purposes, we consider our point sets to already be aligned via a rigid transformation based upon a distinctive rigid pattern of markers organized on the back of the user's hand. Since we are dealing with a small number of points, some of which may be missing or occluded, registration under normal assumptions could lead to spurious alignments, such as the knuckles of one posture being aligned with the fingertips of another.

In contrast to many traditional gesture recognition and associated computer vision systems, we consider especially gestures performed in a 3D space rather than a 2D space perpendicular to the camera. Our main contribution is an analysis and comparison of various methods for the classification of relatively sparse, aligned, unlabeled point sets of variable size. We assume that there is no further context for each point set beyond the information contained in the positions themselves. Any system capable of generating positions corresponding to landmarks (e.g. fingertips), especially with respect to some standard reference frame, should benefit from our analysis. In our case, the positions are generated via reflective markers in a motion capture camera system (see Figure 1).

II. Related Work

Gesture recognition, as a means of human-computer interaction, provides an intuitive and effective interface for user control, offering the ability to perform complicated tasks with minimal effort. A significant amount of research involving gestures has been performed in the past two decades with many methods and solutions offered [3] [4]. Hand gesture recognition is an especially appealing branch of the gesture recognition field because it can offer a more tantalizing avenue for the average end-user, even if only for the visceral thrill of execution.

A static gesture, or posture, is one in which the hand remains still in a certain pose, such as holding a closed fist, whereas a dynamic gesture involves motion of the hand, arm, or fingers, such as pointing or waving. Bhuyan et al. [5] use a finite state machine with fuzzy logic to segment continuous gestures from video streams and present an integrated system for recognition of various static and dynamic gestures. Hand gesture recognition is inherently interactive, providing a wide range of applications including virtual reality and games [6], robot control [7] [8], and interactive sign language [9]. Posture recognition is an integral component of gesture recognition. A system should generally ensure that the user's hand is not relaxed and is making the correct shape before positively interpreting the motion.

978-1-4799-3840-7/14/$31.00 ©2014 IEEE

There are many different methods by which hand features can be measured. Gloves are frequently used; Ceruti et al. [10] use wireless magnetic sensors embedded in a glove to detect finger motion and interpret a Braille-like binary code for communication. Vision-based approaches [11] are of particular interest as they do not require any peripheral accessories other than the camera or equivalent sensing device. For example, the Microsoft Kinect has gained prominence as a hand and arm (a subset of its capabilities) gesture recognition platform. However, it currently lacks the precision to model individual fingers at a significant distance and is thus confined to a catalog of broad, sweeping gestures of the body at large or relatively simple postures such as an open versus closed hand [6] [12]. The relatively new Leap Motion Controller offers such a peripheral-free interaction system in a limited 3D space, but its detection currently suffers from some notable limitations. In particular, the Leap Motion Controller is incapable of recognizing a fist, and touching or crossing fingers can lead to spurious approximations of the hand's pose [13]. Wang et al. [14] offer an alternative vision based approach that is capable of detecting a limited class of pinching gestures for 3D CAD applications.

Motion capture cameras are commonly used to model and record human motion for animation in movies and video games. Several others have used markers for gesture recognition tasks. Chang et al. [15] use supervised feature selection techniques to discover a reduced marker set sufficient for classifying certain classes of grasp gestures and use a similar method to our own in determining a local reference frame for the hand. Liu and McMillan [16] propose a method to estimate missing marker positions during motion from a Random Forest based on local linear kinematic models, which is related to previous work that focused on accurately estimating the motion itself with limited markers [17]. Their method is not directly applicable to our data since we do not consider motion. Martin et al. [18] use a similar camera system to our own in order to recognize specific user actions, such as lifting and tipping a carton of milk or writing, via a combination of vector quantization and dynamic time warping. Lee and Tsai [9] also use a Vicon camera system with neural networks that are trained to completion to recognize 20 Taiwanese sign language static gestures. Both of these works are distinguished from our own in that they do not deal with anonymous markers but with labeled entities; i.e. it was known prior to classification which marker corresponded to the thumb or other location. Martin et al. [18] employed Vicon Nexus software to define a skeletal model of the user's hand, although it is not clear how Lee and Tsai [9] accomplished the labeling. At no point in time do we know which marker is the "thumb" or something similar. Note that in our terminology, gestures such as "thumbs up" or "thumbs down" (used by Song et al. [11]) are considered synonymous as they correspond to the same posture. By including details such as orientation after posture recognition, one can make more refined distinctions (e.g. thumbs up at θ degrees).

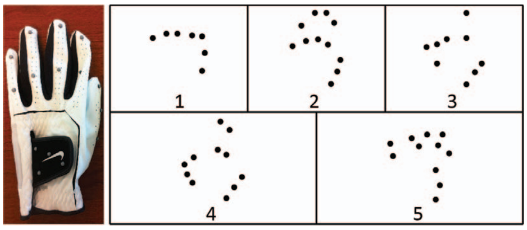

Fig. 1. The glove used to capture data along with a sample from each class of posture projected onto the local XY plane. The classes are fist (1), stop (2), point with one finger (3), point with two fingers (4), and grab (5).

III. Methodology

In this section, we describe our dataset and the classification algorithms evaluated with it. To our knowledge, there is no public dataset directly related to our purpose, i.e. a dataset comprised of instances of small unordered point sets, especially for 3D hand gesture or posture recognition. Therefore, we have produced our own. Several classification algorithms were evaluated on both the raw data (unordered positions) as well as on certain feature transformations.

A. Data

The dataset is comprised of 5 hand postures captured for 12 users using a Vicon motion capture camera system and a glove with attached infrared markers on certain joints. A rigid pattern on the back of the glove was used to establish a translation and rotation invariant local coordinate system. The five postures captured were fist, pointing with one finger, pointing with two fingers, stop (hand flat), and grab (fingers curled) (Figure 1). Several hundred instances of each posture were captured as part of intermittent streams where the user held the posture for a short time. Instances were preprocessed by removing all markers more than 200 mm from the origin and transforming to local coordinates. Since each instance is a variable-size (due to occlusion) unordered set of 3D points, multiple derivative datasets were created to address the lack of structure.

1) Raw: This dataset is comprised of simply the instances with the minimal preprocessing described above.

2) Aggregate: We extract aggregate features that do not depend on the points' order. In particular, the following aggregate features were considered: number of markers, mean, eigenvalues and vectors of the points' covariance matrix, and the dimensions of the axis-aligned minimum bounding box centered on the mean. The expectation is that aggregate features will suffice as long as marker occlusion is not too severe, at which stage more locally sensitive features may be beneficial.

3) Grid Transformation: Although one could rasterize the space, the resolution of the rasterization would likely be prohibitive in terms of associated time and space constraints. As an alternative, we used a low-resolution pseudo-rasterization based upon a limited 3D grid of overlapping spheres. Cubes or diamonds could have alternatively been used by changing the spheres' associated L^p metric, but these were not ultimately considered. Each sphere contributes one feature to a transformed instance, recording the presence of markers within its boundary according to some function of their Euclidean distance to the sphere's center. In this manner, we impose a spatially relevant order on the raw features. Step, linear, and Gaussian functions $f_i(x)$ of a marker's distance x from the center of a sphere with radius r were considered:

$$f_1(x) = \begin{cases} 1 & \text{if } x < r, \\ 0 & \text{otherwise,} \end{cases} \tag{1}$$

$$f_2(x) = \max\left[1 - \frac{x}{r}, 0\right], \tag{2}$$

$$f_3(x) = \exp\left[-\frac{\sigma^2}{2}\left(\frac{x}{r}\right)^2\right], \tag{3}$$

where σ is the number of standard deviations (compared to standard normal) within the sphere. Each function is scaled so that it has a value of 1 for $x = 0$ and a value of 0 (or near 0 in the case of the Gaussian with $\sigma = 3$) for $x \geq r$. In a sense, the spheres are like neurons in a neural network whose activations are triggered by the markers. The activations caused by multiple markers in the same sphere are simply aggregated in a summation.

We first determined a box in which the user's hand was expected to lie based upon the mean position plus or minus two standard deviations. Supposing there are m spheres per dimension, the spheres are scaled and arranged such that they form a regular, densely packed grid spanning the internal volume of the box. Their radii are then uniformly scaled by an integer multiple r_s such that the entire internal volume is covered. The advantage of letting the spheres overlap lies in the implicit creation of extra detection regions according to their intersections. We considered 36 transformations based upon the following options: $m \in \{3, 4, 5, 6\}$, $r_s \in \{2, 3, 4\}$, and $f_i(x)$ with $i \in \{1, 2, 3\}$. See Figure 2 for a visualization.

B. Classifiers

Our classifiers are split into multiple categories based upon the associated type of dataset. We will list raw data classifiers first, which require extra explanation, before providing the traditional classifiers, e.g. support vector machines (SVM), considered for use with the aggregate and transformed data. First, we briefly describe two tools used thoroughly in the raw classifiers, Gaussian mixture models (GMMs) and minimum-cost matchings.

A GMM is a collection of n multivariate Gaussian distributions, or components, used to estimate an arbitrary probability density distribution. Each component c_k is defined by a mean $\mu_k \in \mathbb{R}^d$ and a $d \times d$ covariance matrix Σ_k with probability density function

$$\begin{aligned} f_k(\mathbf{x}) &= P(\mathbf{x}|c_k) \\ &= \frac{1}{\sqrt{(2\pi)^d|\Sigma_k|}} \exp\left[-\frac{1}{2}(\mathbf{x}-\mu_k)^T\Sigma_k^{-1}(\mathbf{x}-\mu_k)\right], \end{aligned} \tag{4}$$

where $\mathbf{x} \in \mathbb{R}^d$. Each component c_k also has a mixture gain or weight π_k denoting its contribution to the overall distribution.

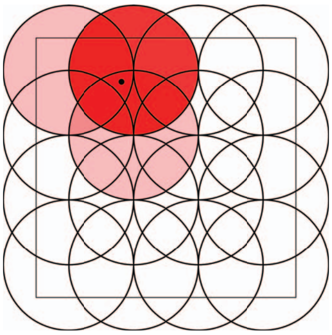

Fig. 2. A 2D grid transformation with $m = 4$, $r_s = 2$, and $i = 2$. The opacity of each sphere is proportional to its activation by the top-left marker.

The weights are normalized such that

$$\sum_{k=1}^{n} \pi_k = 1. \tag{5}$$

The probability of a vector $\mathbf{x}$ given the entire GMM is defined as

$$\mathcal{M}(\mathbf{x}) = \sum_{k=1}^{n} \pi_k f_k(\mathbf{x}). \tag{6}$$

The learning of GMMs is a difficult problem with no known globally optimal solution.

A minimum-cost matching is a solution to the assignment problem [19]. In our case, we wish to assign the points of one instance to the points of another (or to the components of a GMM) such that the summation of costs over all matched pairs is minimized. In this scenario, a GMM component's distribution describes the region in which a marker (such as the tip of the thumb) is expected to lie. Each component represents the expected location of a certain marker. The mixture gains represent the probability of each component being present at all in a given instance. A given GMM approximates the shape of a certain posture, and therefore we construct one per class.

Five cost functions for matching component c_k to the j-th position $\mathbf{x_j}$ of an instance were considered. The first, $C_1(c_k, \mathbf{x_j})$, is simply the Euclidean distance between the point and the mean:

$$C_1(c_k, \mathbf{x_j}) = ||\mathbf{x_j} - \mu_k||_2. \tag{7}$$

The other cost functions measure the probability of observing $\mathbf{x_j}$ independently of other components and factors. The first of these is a normalized version of the component's probability density function:

$$C_2(c_k, \mathbf{x_j}) = -\ln\left(f_k(\mathbf{x_j})\sqrt{(2\pi)^3|\Sigma_k|}\right), \tag{8}$$

where the negative logarithm is taken so that the minimum-cost matching will maximize the product of independent probabilities. Similarly,

$$C_3(c_k, \mathbf{x_j}) = -\ln\left(1 - \chi^2\left((\mathbf{x_j}-\mu_k)^T\Sigma_k^{-1}(\mathbf{x_j}-\mu_k)\right)\right), \tag{9}$$

where $\chi^2(z)$ is the cumulative distribution function of a chi-squared variable of degree 3. Note that z in $C_3(c_k, \mathbf{x_j})$ is the square of the Mahalanobis distance, which has a chi-squared distribution [20]. To produce the last two cost functions, C_2 and C_3 are augmented in the following manner:

$$C_4(c_k, \mathbf{x_j}) = C_2(c_k, \mathbf{x_j}) - \ln \pi_k, \tag{10}$$
$$C_5(c_k, \mathbf{x_j}) = C_3(c_k, \mathbf{x_j}) - \ln \pi_k, \tag{11}$$

effectively accounting for which components are more likely to appear.

For classification based upon a matched GMM, we choose the class corresponding to the GMM with the minimum average-cost (per component) matching. An unmatched component c_k is given a cost of $-\ln(1 - \pi_k)$, the probability of it being absent in a given instance.

1) Greedy GMM: The standard algorithm for computing a GMM with a fixed number of components is the Expectation-Maximization algorithm. Since we do not necessarily know how many components to expect, we use the greedy algorithm of Verbeek et al. [21]. The GMMs are constructed using the greedy algorithm on the union of all training sets I. For unmatched classification, we treat each instance I as a small standalone dataset and classify it as the GMM with the highest log-likelihood

$$\mathcal{L}(I) = \sum_{\mathbf{x} \in I} \ln \mathcal{M}(\mathbf{x}). \tag{12}$$

The classification corresponds to which model's parameters are more likely given the data in I. A matched version uses the same greedy algorithm for construction with C_4 or C_5 for classification.

2) Heuristic GMM: We also explore alternative heuristics employing minimum-cost matchings for constructing the mixtures. The underlying motive for the heuristic GMM is to produce a pseudo-naive Bayes classifier where each GMM component contributes an independent observation. Note, however, that we do not produce a GMM in the strict sense of the definition (the mixture gains will not add up to 1). However, GMMs do provide a useful vocabulary with which to discuss the classifier. The algorithm in Figure 3 was used to produce GMMs that respect the constraint where two markers of the same instance cannot belong to the same component. One of C_1, C_4, or C_5 is used for classification. If C_1 is chosen, then we ignore the cost of an unmatched component as we are not performing a probabilistic classification. Let $O(C_i, C_j)$ denote a heuristic GMM built with option O and cost function C_i that classifies according to C_j.

3) Raw Nearest Neighbor: We use the minimum mapping metric [22] and the k-nearest neighbor (k-NN) graph generation and search schemes of Dong et al. [23] and Hajebi et al. [24]. The minimum mapping metric is similar to a normalized Earth Mover's Distance [25], but remains a metric on sets of unequal size. As such, it measures the cost of transforming one set into the other via a minimum-cost deformation. A majority vote among the 6 nearest neighbors is used to classify a given query instance, with ties broken by the query's minimum average distance per class of neighbor.

```
procedure matchInstances($G$, $I$, $C$)
Given: GMM $G$, set of instances $I$, cost function $C$
for all instances $i$ in $I$ do
  Match points of $i$ to components of $G$ according to $C$
  for all points $p$ of $i$ do
    if $p$ is matched to component $c$ then
      Merge $p$ into $c$; update mean and covariance
    else
      Add component to $G$ with mean $p$
    end if
  end for
end for

procedure rematchInstances($G$, $I$, $C$)
Given: GMM $G$, set of previously matched instances $I$, cost
function $C$
for all instances $i$ in $I$ do
  Remove points of $i$ from prior matched components
  Rematch and merge points of $i$ into components of $G$
end for

procedure train($I$, $C \in \{C_1, C_2, C_3\}$, $O \in \{R, E\}$)
Given: Set of instances $I$, cost function $C$, option $O$
Output: GMM $G$ that approximates $I$
Initialize $G$ with 0 components
matchInstances($G$, $I$, $C$)
while $G$ is not converged do
  Randomly permute $I$
  if $O$ equals $R$ then
    rematchInstances($G$, $I$, $C$)
  else
    matchInstances($G$, $I$, $C$)
  end if
end while
Set mixture gains to percentage of $I$ containing matches
```

Fig. 3. The algorithm and subprocedures used to train heuristic GMMs. Convergence depends upon O. If R is used, convergence occurs when the number of markers rematched to a different component drops below a threshold. Otherwise, convergence occurs when the average matching cost under C stabilizes. A maximum number of iterations is allowed before convergence.

4) Traditional: Six traditional classifiers are considered for the aggregate and grid transformed datasets: naive Bayes, Bayesian networks, multilayer perceptrons (MLPs), SVMs, random forests, and k-NN (with $k = 6$). The implementation and testing of these classifiers is provided by WEKA [26]. Grid transformed classifiers employ feature selection (FS) to both reduce processing time and improve accuracy since many of the spheres in the grid hardly ever contain a marker. Aggregate classifiers did not necessarily use FS. Generally, with the exception of k in k-NN, we let WEKA choose default classification parameters.

C. Evaluation

Due to the streaming nature of the data capture, it is likely that for an instance of a given user there will be a duplicate or near duplicate within the user's dataset. Therefore, we adopted a leave-one-user-out evaluation strategy. In addition, this strategy allows us to measure the ability of a given classifier to generalize to users it has not seen before, just as it would need to do in practice.

We found it prohibitive in terms of time to consider every possible combination of grid transformed dataset, traditional classifier, and left out user. Thus, we opted to first select the "best" on-average classifier and dataset combination via a reduced user set of 4 randomly selected users and 12 randomly selected transformations. The selected classifier would then be compared against the raw and aggregate classifiers on the remaining 8 users.

We used *balanced error rate (BER)* as our primary metric for evaluating the performance of a classifier on c classes with confusion matrix A:

$$\text{BER} = 1 - \frac{1}{c}\sum_{i=1}^{c} \frac{A_{ii}}{\sum_{j=1}^{c} A_{ij}}. \tag{13}$$

BER weights classes equally regardless of their representation in the dataset. If all classes are equally weighted, then it is equivalent to 1 minus the accuracy. Thus, the lower the value of this metric, the better our perceived evaluation of the classifier.

IV. Results

The results for all evaluated classifiers may be found in Table I. Each subsection denotes a type of classifier and is sorted according to average BER. The "best" on-average grid transformed classifier was determined to be the MLP, and it attained its best performance on a transformation with 6 spheres per dimension, each of radius 4, and the linear function $f_2(x)$ (Equation 2). The other traditional classifiers were also evaluated on this transformation. For reference, we included the results of a "Simple" naive Bayes classifier based upon a single feature, the number of visible markers.

The transformed classifiers had a wide range of performance. Even though the MLP achieved the best on-average performance, the k-NN classifier performed better on the chosen final transformation. Note that increasing the number of spheres or sphere radii may yield improved performance. However, this increased transformation complexity automatically translates to increased model complexity, which potentially complicates training and increases the risk of overfitting.

Aggregate classifiers performed fairly similarly to each other, exhibiting fairly balanced performance across the different users. On average, though, they were generally on the higher end in terms of BER. FS did not generally yield improvement in average BER, which is not particularly surprising given the relatively small number of aggregate features. Deviation in BER across users, on the other hand, was generally reduced by FS. Only the MLP and Bayesian network noticeably benefited from attribute selection. Many of the aggregate classifiers had relatively low deviation, reflective of the smooth nature of their features (i.e. they are not prone to overfitting).

TABLE I
The average BERs per user left out and corresponding standard deviations for the classifiers tested, organized by type of classifier. Lower BER is better.

Classifier	**Type**	**BER**
Unmatched Greedy	Raw	0.416 ± 0.183
Matched Greedy (C_5)	Raw	0.681 ± 0.155
Matched Greedy (C_4)	Raw	0.719 ± 0.134
$R(C_1, C_5)$	Raw	0.192 ± 0.180
$R(C_1, C_4)$	Raw	0.194 ± 0.178
$R(C_2, C_4)$	Raw	0.197 ± 0.192
$E(C_2, C_4)$	Raw	0.199 ± 0.203
$E(C_1, C_5)$	Raw	0.203 ± 0.179
$E(C_3, C_5)$	Raw	0.203 ± 0.190
$R(C_3, C_5)$	Raw	0.216 ± 0.227
$E(C_1, C_4)$	Raw	0.217 ± 0.169
$R(C_1, C_1)$	Raw	0.375 ± 0.211
$E(C_1, C_1)$	Raw	0.383 ± 0.211
k-NN	Raw	0.214 ± 0.089
SVM	Aggregate	0.216 ± 0.136
Random Forest	Aggregate	0.221 ± 0.156
SVM (FS)	Aggregate	0.232 ± 0.098
MLP (FS)	Aggregate	0.248 ± 0.098
Naive Bayes	Aggregate	0.273 ± 0.117
MLP	Aggregate	0.289 ± 0.128
Random Forest (FS)	Aggregate	0.292 ± 0.148
k-NN	Aggregate	0.300 ± 0.165
k-NN (FS)	Aggregate	0.301 ± 0.142
Naive Bayes (FS)	Aggregate	0.303 ± 0.202
Bayes-Net (FS)	Aggregate	0.352 ± 0.101
Bayes-Net	Aggregate	0.421 ± 0.187
k-NN	Transformed	0.158 ± 0.152
MLP	Transformed	0.183 ± 0.168
SVM	Transformed	0.204 ± 0.155
Random Forest	Transformed	0.241 ± 0.151
Bayes-Net	Transformed	0.353 ± 0.154
Naive Bayes	Transformed	0.375 ± 0.181
Simple	N/A	0.540 ± 0.123

The matched pseudo-GMMs built using the provided algorithm performed best on average among all raw classifiers, in particular the variants that trained with the C_1 cost function and classified with C_4 or C_5. The reasoning for this is not completely clear; perhaps using the incomplete covariance as in C_2 or C_3 leads to a feedback loop that augments the initial error. On the other hand, ignoring covariance entirely by building and classifying with C_1 yielded very poor results. Note, however, that the raw nearest neighbor has significantly less deviation among the users despite possessing slightly worse on average performance. This lower deviation indicates that it generalizes more readily. The GMMs are prone to overfitting. The greedy GMMs performed quite poorly, likely due to the implicit assumption that a given marker would appear in roughly the same place for each user. Since marker constraints were ignored, overlapping distributions from each user caused the resulting components to poorly reflect the true distributions of individual markers. This problem was magnified in the matched greedy GMMs, whose components clearly did not represent the expected locations of the markers.

We also note that some users are inherently harder to classify than others, regardless of the chosen algorithm. One user consistently had the highest error when left out, even though their rates of marker occlusion were not particularly abnormal or even above average. We think that this supports the idea that an online learning scheme will inevitably be required for any system that expects users to perform natural gestures rather than those precisely dictated by the system. Regardless of the initial training set, there will likely be outlying users that require the system to adapt automatically or through some form of positive and/or negative feedback.

V. Conclusion

We have demonstrated the performance of several classification algorithms on a variety of data transformations of small unlabeled point sets for 3D hand posture recognition. We found each data transformation to have inherent advantages and disadvantages. Aggregate features lead to classification with reduced deviation but limited peak performance. Raw feature classifiers tended to the extremes in both overall error rate and deviation, likely due to their propensity for overfitting. On the other hand, the training objectives also significantly affected performance, as indicated by the results of the greedy GMMs. Grid transformed classifiers also possessed the potential for overfitting, but were capable of achieving maximum accuracy among the algorithms tested. In designing a classifier, one should find a balance between global (e.g. aggregate) and local (e.g. individual point coordinates) features. Future work will explore online learning schemes and the scalability of some of these same classifiers to unknown users. Some classifiers can be retrained more easily than others. In addition, exploiting context such as that contained in the preceding instances of a stream will also be explored. For example, one could exploit context to track positions from instance to instance and thus derive a correspondence other than that produced by the assignment problem.

Acknowledgment

This research is supported in part by the AFOSR grants FA9550-09-1-0479 and FA9550-09-1-0289.

References

[1] P. J. Besl and N. D. McKay, “A method for registration of 3-d shapes,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239–256, Feb. 1992.

[2] B. Jian and B. Vemuri, “Robust point set registration using gaussian mixture models,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 33, no. 8, pp. 1633–1645, Aug 2011.

[3] S. Mitra and T. Acharya, “Gesture recognition: A survey,” *IEEE Trans. Systems, Man and Cybernetics Part C: Applications and Reviews*, vol. 37, no. 3, pp. 311–324, 2007.

[4] J. P. Wachs, M. Kölsch, H. Stern, and Y. Edan, “Vision-based hand-gesture applications,” *Communications of the ACM*, vol. 54, no. 2, pp. 60–71, 2011.

[5] M. K. Bhuyan, D. Ghosh, and P. K. Bora, “Hand motion tracking and trajectory matching for dynamic hand gesture recognition,” *J. Experimental and Theoretical Artificial Intelligence*, vol. 18, no. 4, pp. 435–447, 2006.

[6] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake, “Real-time human pose recognition in parts from single depth images,” in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*. IEEE, 2011, pp. 1297–1304.

[7] M.-S. Chang and J.-H. Chou, “A robust and friendly humanrobot interface system based on natural human gestures,” *Int’l J. Pattern Recognition and Artificial Intelligence*, vol. 24, no. 06, pp. 847–866, 2010.

[8] J. Maycock, J. Steffen, R. Haschke, and H. Ritter, “Robust tracking of human hand postures for robot teaching,” in *Proc. IEEE/RSJ Int’l Conf. Intelligent Robots and Systems*. IEEE, 2011, pp. 2947–2952.

[9] Y.-H. Lee and C.-Y. Tsai, “Taiwan sign language (tsl) recognition based on 3d data and neural networks,” *Expert Systems with Applications*, vol. 36, no. 2 PART 1, pp. 1123–1128, 2009.

[10] M. G. Ceruti, V. V. Dinh, N. X. Tran, H. Van Phan, L. T. Duffy, T. Ton, G. Leonard, E. Medina, O. Amezcua, S. Fugate, G. J. Rogers, R. Luna, and J. Ellen, “Wireless communication glove apparatus for motion tracking, gesture recognition, data transmission, and reception in extreme environments,” in *Proc. ACM Symp. Applied Computing*. ACM, 2009, pp. 172–176.

[11] Y. Song, D. Demirdjian, and R. Davis, “Continuous body and hand gesture recognition for natural human-computer interaction,” *ACM Trans. Interactive Intelligent Systems*, vol. 2, no. 1, pp. 5:1–5:28, Mar. 2012.

[12] J. Han, L. Shao, D. Xu, and J. Shotton, “Enhanced computer vision with microsoft kinect sensor: A review.” *IEEE Trans. Cybernetics*, vol. 43, no. 5, pp. 1318–1334, 2013.

[13] L. E. Potter, J. Araullo, and L. Carter, “The leap motion controller: A view on sign language,” in *Proc. 25th Australian Computer-Human Interaction Conf.: Augmentation, Application, Innovation, Collaboration*. New York, NY, USA: ACM, 2013, pp. 175–178.

[14] R. Wang, S. Paris, and J. Popović, “6d hands: Markerless hand-tracking for computer aided design,” in *Proc. 24th Annual ACM Symp. User Interface Software and Technology*. New York, NY, USA: ACM, 2011, pp. 549–558.

[15] L. Chang, N. Pollard, T. Mitchell, and E. Xing, “Feature selection for grasp recognition from optical markers,” in *Proc. IEEE/RSJ Int’l Conf. Intelligent Robots and Systems*. IEEE, 29 2007-nov. 2 2007, pp. 2944 –2950.

[16] G. Liu and L. McMillan, “Estimation of missing markers in human motion capture,” *Visual Computer*, vol. 22, no. 9, pp. 721–728, Sep. 2006.

[17] G. Liu, J. Zhang, W. Wang, and L. McMillan, “Human motion estimation from a reduced marker set,” in *Proc. Symp. Interactive 3D Graphics and Games*. New York, NY, USA: ACM, 2006, pp. 35–42.

[18] M. Martin, J. Maycock, F. P. Schmidt, and O. Kramer, “Recognition of manual actions using vector quantization and dynamic time warping,” in *Proc. HAIS*. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 221–228.

[19] J. Edmonds and R. M. Karp, “Theoretical improvements in algorithmic efficiency for network flow problems,” *J. ACM*, vol. 19, no. 2, pp. 248–264, 1972.

[20] G. A. F. Seber, *Multivariate observations*. New York, NY: Wiley, 1984.

[21] J. J. Verbeek, N. Vlassis, and B. Krse, “Efficient greedy learning of gaussian mixture models,” *Neural Computation*, vol. 15, pp. 469–485, 2003.

[22] A. Gardner, J. Kanno, C. A. Duncan, and R. Selmic, “Measuring distance between unordered sets of different sizes,” in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*. IEEE, 2014, pp. 137–143.

[23] W. Dong, C. Moses, and K. Li, “Efficient k-nearest neighbor graph construction for generic similarity measures,” in *Proc. 20th Int’l Conf. World Wide Web*. New York, NY, USA: ACM, 2011, pp. 577–586.

[24] K. Hajebi, Y. Abbasi-Yadkori, H. Shahbazi, and H. Zhang, “Fast approximate nearest-neighbor search with k-nearest neighbor graph,” in *Proc. 22nd Int’l Joint Conf. Artificial Intelligence*. AAAI Press, 2011, pp. 1312–1317.

[25] Y. Rubner, C. Tomasi, and L. J. Guibas, “The earth movers distance as a metric for image retrieval,” *Int’l J. Computer Vision*, vol. 40, p. 2000, 2000.

[26] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, “The weka data mining software: An update,” *SIGKDD Explorations Newsletter*, vol. 11, no. 1, pp. 10–18, Nov. 2009.