Grasp Recognition with Uncalibrated Data Gloves - A Comparison of Classification Methods

Guido Heumer* Heni Ben Amor† Matthias Weber‡ Bernhard Jung§

VR and Multimedia Group
Institute of Informatics
TU Bergakademie Freiberg
Germany

ABSTRACT

This paper presents a comparison of various classification methods for the problem of recognizing grasp types involved in object manipulations performed with a data glove. Conventional wisdom holds that data gloves need calibration in order to obtain accurate results. However, calibration is a time-consuming process, inherently user-specific, and its results are often not perfect. In contrast, the present study aims at evaluating recognition methods that do not require prior calibration of the data glove, by using raw sensor readings as input features and mapping them directly to different categories of hand shapes. An experiment was carried out, where test persons wearing a data glove had to grasp physical objects of different shapes corresponding to the various grasp types of the Schlesinger taxonomy. The collected data was analyzed with 28 classifiers including different types of neural networks, decision trees, Bayes nets, and lazy learners. Each classifier was analyzed in six different settings, representing various application scenarios with differing generalization demands. The results of this work are twofold: (1) We show that a reasonably well to highly reliable recognition of grasp types can be achieved – depending on whether or not the glove user is among those training the classifier – even with uncalibrated data gloves. (2) We identify the best performing classification methods for recognition of various grasp types. To conclude, cumbersome calibration processes before productive usage of data gloves can be spared in many situations.

Keywords: Data Glove, Calibration, Grasp Recognition, Classification Methods

Index Terms: I.3.6 [Computer Graphics]: Methodology and Techniques—Interaction techniques; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—[Virtual Reality]

1 INTRODUCTION

A desirable goal for many applications of immersive VR is the support of natural virtual object manipulations that closely resemble the manipulation of real objects. Natural object manipulations are e.g. fundamental in virtual prototyping for accurate simulation of the operation or assembly of virtual product models [24]. Similarly, the imitation of a VR user's manipulation of virtual objects has been proposed as a means for programming assembly robots by demonstration [1] and for generating virtual character animations that faithfully reproduce the user's interactions with scene objects [13]. To support such natural manipulations, it is crucial that the VR system is able to differentiate between various types of human grasping.

VR-based manipulations of virtual objects are commonly facilitated through data glove-type input devices, such as Immersion's Cyberglove. In order to recognize a user-performed grasp, the sensor readings of the data glove have to be processed, analyzed and matched towards one of a set of known grasp types. Typically, before using the data gloves a time-consuming calibration phase is needed in order to account for differences in hand size and proportion when mapping from raw sensor readings to joint angles of the user's hand. How an optimal calibration can be achieved is still an unsettled question. The more accurate methods rely on external vision systems, which themselves need to be calibrated.

A main motivation for the work described here is to find out whether it is possible to recognize a range of hand shape types during manipulations directly from raw sensor input without the intermediate joint angle representations. If successful, the cumbersome calibration phase could be spared, enabling an immediate productive use of data gloves in immersive VR systems in many situations where reliable classification of hand shapes (rather than exact reconstruction of joint angle values) is sufficient for the application.

A second motivation for this work is to evaluate the performance of different classification methods. The aim is to identify the classifier (or family of classifiers) which is suited best for the problem domain of "grasp recognition from raw data glove sensor data". The reason behind this are the so-called "no free lunch" (NFL) theorems [23]. The NFL theorems state, that averaged over all possible problems, all algorithms perform exactly equal. As a consequence, there can not be one classification technique, which is optimal for all classification tasks. However, when restricting the classification problem to a particular domain, there might well be a classifier (or a set of classifiers) which outperforms all others. This leads to the conclusion that selecting a good classifier should be based on an empirical evaluation in the respective problem domain. Following this reasoning we systematically evaluated several classification techniques for this problem domain. We have experimented with a total of 28 classifiers in various settings to find out which classifiers are suited best for this type of problem. The settings reflect different possible use-cases and application scenarios. In this way, informed decisions can be made about whether or not to use a particular classifier in a given VR scenario.

2 RELATED WORK

Various research in the fields of medicine, robotics, developmental psychology and VR has led to the formulation of grasp taxonomies: categorizations of grasps based on form or function. An early taxonomy is described in Schlesinger's work on constructing artificial hands [20]. He characterized which functionalities in prosthetic hands are needed to grasp certain objects. Building on this work, Taylor and Schwartz [21] defined English names for the most important grasps investigated by Schlesinger: cylindrical, tip, hook, palmar, spherical and lateral grip (see Figure 1 for examples).

*e-mail: guido.heumer@informatik.tu-freiberg.de
†e-mail: amor@informatik.tu-freiberg.de
‡e-mail: matthias.weber@informatik.tu-freiberg.de
§e-mail: jung@informatik.tu-freiberg.de

IEEE Virtual Reality Conference 2007
March 10 - 14, Charlotte, North Carolina, USA
1-4244-0906-3/07/$20.00 ©2007 IEEE

Napier [16] researched the basic task requirements of grasps and differentiated between two basic grasp types: the power grip which clamps an object firmly under usage of the palm and the precision grip where the thumb and other fingers pinch the object. Later, Cutkosky [7] investigated optimal grasp operations in factories and developed a taxonomy for categorizing feasible grasp types in this domain.

In order to enable the computer to recognize and match a user-performed grasp onto a corresponding class from the taxonomy, techniques from the area of pattern classification can be applied. In Friedrich et al. [11] a Neural Network classifier and the Cutkosky taxonomy were used for this purpose, yielding a classification rate of about 90% for grasps performed using a data glove. According to Ekvall and Kragic [9], a Hidden Markov Model (HMM) based method was even able to achieve recognition rates of close to 100% for single user settings. However, recognition rates dropped significantly (to about 70%) for settings with multiple users. In Aleotti and Caselli [1] a nearest neighbor classifier is used in conjunction with heuristic rules. The task of these rules is to disambiguate between similar grasps. In this way, recognition rates of 94% for seen users (users trained with) and 82% for unseen users (users not trained with) were achieved. Applying a classifier to unseen users always bears the risk of significantly lower recognition rates. This stems from the fact that even identical postures can produce different sensor values when the sizes of the subjects' hands vary. One way to tackle this problem is to perform a calibration process as in Kahlesz et al. [14]. However, this process can be complex, time-consuming and in itself error-prone. In a recent paper by Borst and Indugula on realistic virtual grasping [4], it was noticed that even time-consuming calibration procedures do not produce accurate results. Another way to solve this problem is to use classification algorithms which are able to generalize over a large set of users. However, it is still an open question which classification techniques can achieve such generalization as no thorough comparison has been conducted so far. Another interesting question which needs further investigation is the performance of classification techniques in different settings; e.g when new objects are grasped.

In contrast to previous research, the work presented in this paper does not focus on a particular classification algorithm or a particular setting. Instead, we try to compare the performance of a wide range of classifiers in several settings within the domain of grasp classification. Such a comprehensive evaluation enables us to draw various conclusions about the applicability and the success of classification with uncalibrated data gloves.

3 Data Acquisition

In the data acquisition phase of our study, sensor value data was captured from a couple of users, performing all the grasps of Schlesinger's taxonomy [20] on several real objects. After recording the raw data, a first analysis was done on the basis of Sammons mapping of a Self-Organizing Map [15]. The experiment setup and the results of the data analysis are presented after a short illustration of how the hand posture is measured by the type of data glove used.

3.1 Data Glove

The data glove used for recording was a 22-sensor wireless Cyberglove 2 by Immersion, Inc. (see Figure 1). This type of data glove measures hand posture through a number of resistive bend-sensing sensors which are placed in key locations (mostly joint positions) on a stretch fabric glove. Each of the sensors measures its amount of bending around one axis (the flat side) in the form of an 8-bit value between 0 and 255, which is almost linearly proportional to the bend angle.

It is important to note that the measured sensor values *do not* directly represent finger joint angle values. For the mapping from sensor values to actual joint angles, a complex calibration and conversion process is necessary which involves several pitfalls. In the easiest case of measuring the flexion of the interphalangeal joints, a direct linear conversion from one sensor value to a joint angle can be performed. This involves an offset value (sensor value when the joint is considered straight) and a gain factor (multiplicative factor to convert bend value to radians/degrees). Even for this simplest method of mapping, offset and gain values have to be determined for each single sensor in a tedious process. Due to cross-couplings between the sensors, however, more complex forms of calibration are necessary to achieve a satisfactory fidelity. In Kahlesz et al. [14] some recent calibration techniques are summarized. Since our objective was to spare any calibration process, the raw sensor readings, as transmitted by the Cyberglove 2, were used directly as feature vector for hand posture classification.

3.2 Experiment Setup

For each of the six grasp types, four objects of various shapes were grasped. Table 1 lists the objects used for each grasp type, Figure 1 shows pictures of some of these objects.

Grasp Type	Objects
Cylindrical	Bottle, Hammer, Flower Pot, Coffee Jug
Hook	Plastic Case, Toolbox, Backpack, Bag
Lateral	Floppy Disk, Key, ID Card, CD Cover
Palmar	Small Box, Matchbox, Tape Roller, PDA
Spherical	Tennis Ball, Egg-shaped Case, Bowl, Mouse
Tip	Nail, Pencil, Small Eraser, PDA

Table 1: The grasp types and corresponding trial objects.

For each object, two trial sequences were performed. In the first sequence, the object was grasped five times, with the subject's grasping hand starting from a fixed position on the table. During this whole sequence the subject sat at the table. In the second sequence, the object was again grasped five times. This time, however, the hand starting position was varied randomly, as was the object's orientation on the table. For larger objects, like the tool box, the subject was standing during this sequence. The captured data consisted of all the 22 glove sensor values representing the hand posture at the "peak" moment of the grasp – as opposed to a sequence of sensor values of a full grasping movement. This moment means the time the test subject's hand firmly held the object and the fingers were at rest. The peak moment was determined manually by the test subject by pressing a button on the dataglove.

The whole data acquisition process was conducted with six test subjects. Each subject was adult, male and right-handed. In total, 6 (test subjects) $\times$ 24 (objects) $\times$ 10 (grasps per test subject and object) = 1440 data items were collected.

3.3 Data Visualization

Before the classifier evaluation was executed, it proved interesting to first take a "glimpse" at the data. This helped to get an idea of the problem difficulty, to assess the quality of the recorded data or to make first decisions concerning a suitable classifier. Typically, the recorded data is in a higher-dimensional space which makes a human inspection difficult. For inspection and analysis purposes it is more convenient to create a visual representation of the experiment data. This can be achieved through a projection of the high-dimensional pattern space onto two dimensions. Common techniques for such projection tasks are the Self-Organizing Map and Sammons' mapping [15]. Figure 2 shows a projection of our experiment data using a combination of the two techniques. The projection is distance preserving, which means that points which are near to each other in the higher dimensional space will also be

Figure 1: Images of some of the objects used during the grasping experiments with their corresponding grasp types.

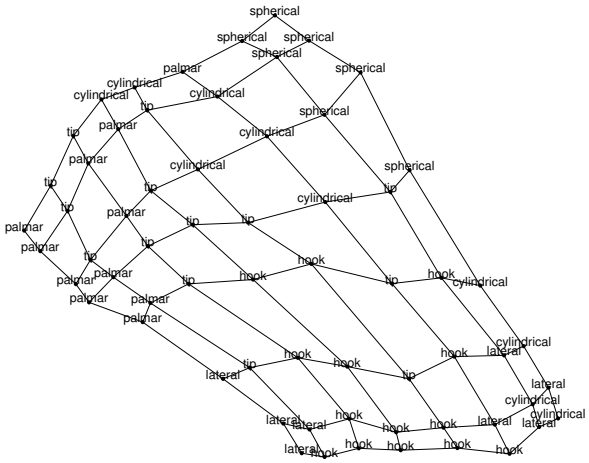

Figure 2: Sammons mapping of a Self-Organizing Map representing the data projected onto a two-dimensional space.

near to each other in the projection. It can be seen that the spherical grasp forms a particular region in the upper part of the map. This region can neatly be separated from regions representing other grasps, which indicates that classification of this grasp type is particularly easy. Although other grasp types also occupy particular regions on the map, they are much more intermixed. This yields complex decision boundaries. For example, the classes "tip" and "palmar" cannot cleanly be separated from each other. What also becomes visible, is that "cylindrical" grasps are scattered throughout the map. This fact is particularly interesting, as it exemplifies the hard separability of the Schlesinger grasps based on hand shape only. It follows from this observation that we can expect classification to be more error-prone for cylindrical grasps.

4 Classifier Evaluation

A set of 28 different classifiers from the freely available Weka data mining software package [22] has been evaluated. These were run in an "out of the box" fashion, i.e. with default settings and without any parameter optimization. The tested classifiers can be broadly divided into five categories – probabilistic methods, function approximators, lazy learners, trees, and rule sets: (1) Probabilistic methods such as the naive Bayes classifier [10] or Bayes nets [6] learn to discriminate between classes by building up probability models of each class. Using Bayesian inference the probability of a new data item belonging to a particular class can be computed. For example, the naive Bayes classifier learns a model of the training data by estimating class probabilities and conditional probabilities of the variables. Together with the Bayes theorem, these values can be used to compute the probability of a data item belonging to a particular class. The only "naive" assumption (hence the name) that is being made, is that all variables of a data item are mutually independent. (2) Function approximators learn the parameters of a function which takes the new data item as an input and returns the class as an output. Well-known representatives of this type of algorithm are Multilayer Perceptrons and Radial Basis Networks [2]. Here, the approximated function is represented by a set of interconnected neurons. The back-propagation algorithm [12] can be used to train such networks in order to minimize the squared error of approximation. (3) Lazy learning techniques [3] postpone any type of learning until a request for classification of a new data item is received. When such a request is received, a database of previously seen examples is searched for a set of examples, which are closest to the new item (w.r.t. a given distance metrics). (4) Tree classifiers, such as decision trees [17], try to break up the classification task into a hierarchy of simple decisions at whose end the final decision determines the class. As the name suggests, this hierarchy has the form of a tree, whose nodes represent local decisions, while leafs represent the classes. (5) Finally, rule induction methods [18] create sets of logical rules for determining the class of a particular item. Ridor [19], or "Ripple Down Rule", is a representative algorithm from this class. Ridor requires that the data is incrementally supplied to the training set. Data items which conflict with previously learned rules are seen as exceptions. These are then treated by patching the rule locally for the particular item.

Another category, namely meta learning techniques [5] were only roughly examined in preliminary tests and are not included in the final results. These are techniques such as boosting or bagging that aim to create powerful classifiers through a combination of several simpler ones. Depending on the algorithm hierarchies, cascades or ensembles of *base* classifiers are used for classification. Since meta classifiers can be built from essentially arbitrary combinations of simpler classifiers, they are inherently more complicated to evaluate and several choices of base classifiers would have to be looked at. This will, however, be subject of future work.

4.1 Design & Method

To determine which classifier is suited best for the domain of classifying raw sensor data, a comprehensive, systematic classifier evaluation was performed. We examined a set of classifiers in 6 different settings, formed by a permutation of the values of two situational variables (see Tables 2 and 3), putting different generalization demands on the classifiers. One variable (objects) determined whether the objects grasped in the test set were seen, i.e. grasp examples with this objects were used for training, versus unseen, where no grasp examples with this object were used during training. Note, that even in the seen case, training and test sets always were disjoint. This means, a grasp example used during training was never used for testing as well. The other variable (user) determined the user group, i.e. which set of users' grasp examples were taken for training. For this variable, three different cases were investigated: *individual* and *group*, where training data of only one user or the full group of users, respectively, were used for training and testing. And a third case, *unseen*, where data of all users except one was used for training, and data of the left-out user was used for testing. The property of disjoint training and test sets also holds true in all three cases.

Value	Meaning	Example Scenario
seen	classifier trained and tested with the same set of objects	applications with a given, fixed set of objects, e.g. a tool set
unseen	classifier trained and tested with different sets of objects	applications where scene objects change or are of modifiable form, e.g. CAD

Table 2: Investigated values of the variable 'objects'.

Value	Meaning	Example Scenario
individual	classifier trained and tested with a specific user	single operator system
group	classifier trained with a group of users and tested with a group member	work group
unseen	classifier trained with several users but tested with a user not in the group	public installation, e.g. game

Table 3: Investigated values of the variable 'user'.

For each of the six settings, several pairs of disjoint training and test sets were generated by splitting the complete data in an adequate way. Each classifier was trained and tested with each pair (or data *split*), and the average rate of correct classifications for each classifier over all these tests was determined. The feature (input) vector for classification consisted of all 22 sensor values which were not weighted. The output of the classifier was an index value, indicating one of the six grasp types. Note, that since all data items were taken from valid examples of the different grasp types, there was no rejection class. The right answer was always one of the six Schlesinger grasp types.

Classifier performance was measured in the percentage of correct classifications. When two classifiers had the same average performance, the classifier with the smaller standard deviation was considered better. Additionally, for each setting, a set of best classifiers was established by selecting all classifiers that performed not significantly different than the best classifier. To determine the significance of differences between classifiers the McNemar's test was used (for an overview on significance tests, see [8]).

For all settings, the number of data splits into test and training set and the respective set sizes per split are summarized in Table 4. Also a rough indication of how the test set was formed is given in the middle column. More detail about the settings and the exact method of how test and training set were generated are given in the following subsections. Readers not interested in this level of detail might want to skip to the presentation of results in section 4.2.

Setting (user, objects)	Data Splitting	Train. Set	Test Set
#1 - individual, seen objects	6 splits of data of one user. test set - two random examples per object.	192	48
#2 - individual, unseen objects	8 splits (2 series of 4) of data of one user. test set - one random object per grasp type.	180	60
#3 - group, seen objects	6 splits - as in #1 but data of all users. test set - two random examples per object and user.	1152	288
#4 - group, unseen objects	8 splits as in #2 but data of all users.	1080	360
#5 - unseen, seen objects	6 splits (one per user). test set - all data of one user.	1200	240
#6 - unseen, unseen objects	12 splits (2 series of 6). test set - from user splits (as in #5) pick one random object per grasp type.	900	60

Table 4: Splitting of data into test and training sets for the different settings.

4.1.1 Individual user, seen objects

Data of only one user is regarded and the same set of objects is used for testing and training. This corresponds to an application, where the system is trained for a specific user (and this user only) and all objects to be interacted with are known in advance. In comparison with conventional (calibrated) classification, this would correspond to a perfect glove calibration being available for a particular user and an additional training session having been performed, where all objects later to be interacted with are trained into the system.

To generate disjoint training and test sets for this setting, all data of one user was taken and split evenly into two sets, so that the respective numbers of examples for each grasp and object stayed the same. Since ten data samples for each object were recorded – five with a fixed starting position and five with a variable starting position – two samples of each object (one with each type of starting position) were chosen for the test set (48 data items), while the others formed the training set (192 data items). Overall, six splits were generated in this way, by randomly chosing the test items.

4.1.2 Individual user, unseen objects

Again, data of only one user is regarded, however tests were always performed with unseen objects, i.e. no data examples of the object used for the tests have been used for training. This corresponds to an application, where the system was trained for a specific user, however the objects used during the interaction are not previously known.

Training and test sets were generated, by randomly choosing one object for each grasp type and using the examples of these object as test set, whereas the data of the other objects was used as training set. This way, for each user a series of eight splits was generated, so that each object of one grasp type was used exactly twice for testing. The training sets consisted of 180 data items, whereas the test sets were 60 items large. The combinations between grasp types, i.e.

which object of each grasp type was chosen, were random. It was ensured, however, that each permutation only occurred once.

4.1.3 Seen group of users, seen objects

Similar to 4.1.1 (individual user, seen objects), however data examples of all users were used for training and testing. This corresponds to an application that is set up to work with a certain group of users and the objects used for interaction are known in advance. Note that this setting, similar to the settings below, is already beyond the scope of calibration-based approaches as these require system knowledge of individual users.

Data splitting was done as in the individual user case, but with data of all users instead of one. Additionally, it was made sure that the same number of examples from each user was chosen. For each object and user combination two examples were chosen for the test set (288 data items), while the remaining examples formed the training set (1152 data items).

4.1.4 Seen group of users, unseen objects

This setting is similar to 4.1.2 (individual user, unseen objects). However, data of all users were used for training and testing instead of data from just one user. This corresponds to an application that is set up to work with a certain group of users and the objects used for interaction are not known in advance.

Training and test sets were generated by creating eight splits, where in each split data of one randomly picked object (per grasp type) forms the test set (360 data items), whereas data from the remaining objects composes the training set (1080 data items). Again, it was made sure that no permutation was repeated and that each object was part of the test sets exactly twice.

4.1.5 Unseen users, seen objects

In this setting, no data of the test user was contained in the training set, i.e. the classifier was not trained with data from the test user. This corresponds to an application, where the system was trained with data from a group of users and another (previously unseen user) then uses the system, for example in public installations, etc. All objects used during tests were seen before by the classifier (grasped by other users) during training, i.e. for this type of application the objects of the interaction need to be known in advance.

Here, for each user, a pair of datasets was generated where the training set contained sensor data from all users except this one (1200 data items), and the test set consisted of all data from this user (240 data items). Since data was acquired from six different users, this resulted in six different disjoint splits.

4.1.6 Unseen users, unseen objects

In this setting neither the objects nor the user involved in testing were seen by the classifier during training. This corresponds to applications where the users and interaction objects are not known in advance. This setting puts high demands on the classifier's generalization capabilities, but can satisfy the broadest area of use cases.

For generating test sets and training sets the data was first split into disjoint sets for each user as in 4.1.5. Then, for each of these user splits two random object splits were generated, where for each grasp type one object was picked. Data from this object was removed from the training sets, whereas only the data of this object remained in the test set. This resulted in twelve data splits overall with a test set size of 60 data items and a training set size of 900 data items.

4.2 Results

For every evaluated classifier in each data split of each setting, the percentage of correctly classified examples has been determined. Due to space limitations this is too much information to be presented here. Hence, the results have been summarized by determining average classification rates for each setting and the corresponding standard deviation, see Table 5. Each setting is represented by two columns (average and standard deviation). Following the cross validation results, in the next two columns are the average performance over all settings and the standard deviation of this total average. In the final column, the average run time per test of each classification algorithm is given in milliseconds. This value is of course dependent on the used hardware and for this reason is only to be seen as a relative comparison between the several algorithms. Grey table cells indicate for each setting the classifier with the highest accuracy; table cells shaded in light gray indicate classification methods whose accuracies vary only insignificantly (according to a McNemar's test) from the best performing classifier. The interested reader can find the complete results of the study as well as the captured data on the Web under `http://vr.tu-freiberg.de/grasping/`.

In the following, the results for each of the settings are summarized:

- **individual user, seen objects** – With 99.48% achieved by the Kstar algorithm, a highly reliable classification rate can be reached for the case where the objects grasped are known in advance and the user trained the system individually. A not significantly worse performance can be reached with IB1, RBF Network, Multilayer Perceptron, LMT, Simple Logistic, NNge, and SMO. Since KStar has a relatively long runtime, in more time critical applications IB1 would be the next-best choice or, if an even shorter runtime is needed, Multilayer Perceptron.

- **individual user, unseen objects** – In the case, where the objects grasped are not known (and trained in) in advance, more generalization ability is needed, and the classification rate drops down to 83.82%. The best classification rate was achieved with SMO, which also has a relatively short runtime. Not significantly worse performed IB1 and Multilayer Perceptron.

- **group of users, seen objects** – In the case, where a whole group of users trained the system and an unspecified member of the group uses it, a classification rate of 99.07%, almost as good as for the single user case, can be achieved. Best performed IB1, followed by KStar and Multilayer Perceptron, which would be the best choice, if a short test runtime is important.

- **group of users, unseen objects** – Again, for unseen objects the classification rate drops, in this case to 84.62%. The best classifier in this setting clearly was IB1 with all other classifiers performing significantly worse.

- **unseen users, seen objects** – For the case, where the user is unseen to the system but the objects are known in advance, a similar classification rate as for the unseen objects cases is achieved. With 81.74% SMO performed best, followed by the not significantly worse Multilayer Perceptron, IB1, Random Forest and LMT classifiers.

- **unseen users, unseen objects** – In this case, where the user as well as the objects are unseen, the classification rate drops further down to 71.67%, achieved by the SMO classifier. This reflects the rather high demand on generalization abilities of the classifiers. Similar performance was achieved by Multilayer Perceptron, IB1, Simple Logistic and RBF Network classifiers.

Classifier	Classifier Category	Individual User, Seen Object		Individual User, Unseen Object		Group, Seen Object		Group, Unseen Object		Unseen User, Seen Object		Unseen User, Unseen Object		Cross Validation	Total		Runtime
		Avg.	Stdev.	Avg.	Stdev.	Avg.	Stdev.	Avg.	Stdev.	Avg.	Stdev.	Avg.	Stdev.	Avg.	Avg.	Stdev.	[ms/test]
IB1	lazy	99.13	0.72	83.68	5.12	99.07	0.72	84.62	4.99	80.07	6.26	68.06	11.72	98.61	87.61	11.89	2.25662
MultilayerPerceptron	functions	98.38	0.9	82.4	5.27	96.88	0.82	80.24	4.45	81.04	3.69	71.53	11.02	97.15	86.8	10.58	0.06072
SMO	functions	97.11	1.56	83.82	5.01	93.64	0.84	81.74	4.48	81.74	7.17	71.67	12.81	93.54	86.18	8.99	0.07422
KStar	lazy	99.48	0.78	79.66	6.74	98.67	0.89	81.81	3.04	77.29	5.73	64.86	11.31	98.54	85.76	13.41	29.62138
LMT	trees	97.28	1.11	76.98	3.09	95.02	0.78	77.12	5.71	79.17	4.53	63.47	14.41	95.07	83.44	12.65	0.09188
SimpleLogistic	functions	97.28	1.11	76.53	2.73	93.81	0.97	79.2	5.22	76.53	7.72	67.22	15.49	92.5	83.3	11.24	0.03572
RandomForest	trees	96.99	1.18	71.98	7.06	96.82	0.86	74.34	5.89	79.86	3.89	66.11	8.86	96.74	83.26	13.33	0.03631
RBFNetwork	functions	98.56	1.08	70.49	8.93	91.61	1.69	77.47	5.27	76.32	11.6	67.08	16.13	90.35	81.7	11.86	0.14287
Logistic	functions	96.01	1.44	72.92	3.76	92.3	1.23	77.71	4.19	73.89	8.26	64.31	12.15	92.71	81.41	12.21	0.04405
NNge	rules	97.22	1.2	68.13	5.06	92.01	2	67.92	8.03	71.53	8.66	57.92	14.41	92.15	78.13	15.33	0.26511
PART	rules	91.55	2.72	66.6	5.93	92.48	1.22	67.29	4.58	73.2	11.38	55.28	13.54	91.74	76.88	15.04	0.03631
J48	trees	93	3.17	66.7	7.9	90.45	1.66	68.44	6.52	69.24	10.43	59.31	14.55	90.83	76.85	14.03	0.03492
NBTree	trees	94.74	2.63	66.04	8.09	91.26	1.06	69.76	5.17	63.13	11.41	52.92	8.77	91.18	75.58	16.58	0.09505
BayesNet	bayes	95.55	2.77	68.06	5.82	85.42	0.76	63.3	6.14	69.72	10.01	61.11	9.17	84.03	75.31	13.02	0.08255
Ridor	rules	90.97	2.62	64.48	5.6	89.18	2.52	66.74	5.55	66.53	8.75	54.03	14.27	87.64	74.22	14.73	0.02937
REPTree	trees	88.66	2.9	65.49	4.6	87.15	1.26	64.58	5.61	65	9.03	58.19	12.5	88.13	73.89	13.41	0.01409
NaiveBayes	bayes	93.69	3.35	71.01	4.84	76.62	0.61	62.71	7.99	68.96	13.42	61.81	11.38	76.32	73.02	10.83	0.23733
JRip	rules	86	2.81	58.48	3.15	90.45	1.4	65.63	4.14	64.31	11.83	56.81	11.58	89.31	73	14.96	0.01607
NaiveBayesUpdateable	bayes	93.64	3.28	70.8	4.56	76.62	0.68	62.33	8.18	69.17	13.62	61.81	11.51	76.32	72.96	10.87	0.27007
NaiveBayesSimple	bayes	93.23	3.34	70.73	4.36	76.62	0.68	62.33	8.18	69.17	13.62	61.95	11.76	76.25	72.9	10.71	0.07322
NaiveBayesMultinomial	bayes	84.55	5.04	65.28	4.12	72.57	0.69	58.33	5.61	66.32	14.63	58.06	13.76	72.29	68.2	9.27	0.03651
RandomTree	trees	84.96	2.96	56.56	6.31	83.57	1.58	56.6	6.34	58.4	8.7	46.25	13.47	82.22	66.94	16.07	0.02738
LWL	lazy	78.88	5.76	63.79	3.51	65.86	1.43	55.31	6.99	59.86	8.27	51.94	12.77	65	62.95	8.72	21.36623
ComplementNaiveBayes	bayes	71.99	10.03	60.28	7.85	63.43	0.9	53.23	6.35	60.07	8.77	55	12.93	63.13	61.02	6.18	0.03631
DecisionTable	rules	80.03	3.28	46.7	6.78	76.39	2.17	52.05	7.27	50.62	6.89	41.11	14.98	76.53	60.49	16.46	0.04068
OneR	rules	53.36	5.92	40.9	6.72	42.71	1.32	30.04	5.99	32.71	7.41	28.75	9.67	43.89	38.91	8.87	0.02600
DecisionStump	trees	32.7	0.56	30.35	0.94	31.89	1.17	31.28	2.24	29.72	4.16	28.61	7.27	32.85	31.06	1.58	0.01449
ConjunctiveRule	rules	32.87	0.57	29.48	1.11	32.12	0.82	29.62	3.91	29.31	2.48	30.42	6.44	32.36	30.88	1.52	0.02004

Table 5: Results of the classifier evaluation with best classifier in setting and not significantly different and therefore additional best classifiers .

To obtain an additional measure for classifier performance, a 10-fold cross-validation on the complete data set was performed. To cross-validate a data set, it is split in k (where k is 10 in this case) subsets. Then, k tests are conducted where the k-th subset is used as the test set for the classifier and the other k-1 subsets are used as the training data. The total classification rate is then computed as the average of the classification rates of all k tests.

In the column labeled "Total", the average value of the average performances in the different settings is denoted. This is an indication of how well a classifier performs overall, hence the table has been sorted by this value. The IB1 algorithm leads the table with 87.61% classification rate. The total standard deviation indicates how strongly classifier performance varies over the different settings. Note, that this is not the average of the standard deviations for the various settings. The average performances of the best classifiers for each setting are also displayed comparatively in Figure 3.

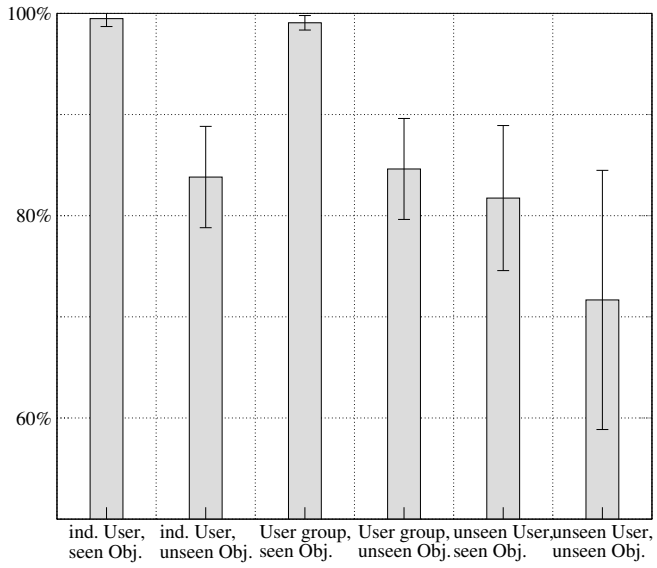

Figure 3: Comparison of average performance of best classifiers for each setting

Average classifier runtimes (per test) have been determined by summarizing the runtimes for all tests and dividing them by the number of tests. They are given in the last column. As can be seen, most runtimes stay in the same order of magnitude. The only exceptions are the lazy evaluators, which have a relatively long runtime, since a lot of training examples need to be considered during the tests. This runtime difference will also further increase with larger training set sizes.

For further analysis, the confusion matrix of all classification results for the best classifier IB1 was investigated. Each entry in this matrix shows the number of recognized categories when a certain grasp is shown to the classifier. For a better understanding of the matrix, these numbers were normalized by dividing by the total number of examples for a certain grasp type. The resulting confusion matrix is presented in Table 6.

The entries in the diagonal of the matrix represent the percentage of which grasps were identified correctly. Since, as has been shown, the IB1 classifier performs quite well, these values are reasonably high. The other values in the matrix describe the percentage of misclassifications between the shown and classified grasp category. They indicate probable difficulties when distinguishing between certain grasps. As predicted by the Sammon's mapping in Section 3.3 the distinction between cylindrical and hook grasps as well as between tip and palmar grasps is problematic. In contrast, spherical and lateral are very well distinguishable from all the other grasp types.

	tip	cyl	sph	pal	hook	lat
tip	0.9698	0.0030	0.0006	0.0214	0.0038	0.0015
cyl	0.0077	0.9329	0.0106	0.0086	0.0375	0.0028
sph	0.0042	0.0032	0.9890	0.0033	0.0000	0.0002
pal	0.0205	0.0024	0.0006	0.9765	0.0000	0.0001
hook	0.0012	0.0176	0.0004	0.0000	0.9732	0.0076
lat	0.0011	0.0002	0.0000	0.0002	0.0028	0.9957

Table 6: The normalized overall confusion matrix for IB1. (Row = shown example, Column = classified as; column names are abbreviations of tip, cylindrical, spherical, palmar, hook, lateral)

4.3 Discussion

It has been shown that grasp recognition based on uncalibrated data glove sensor input can be performed in a very reliable way in specific scenarios, where the group of users that uses the system and the objects that are used during interaction are known in advance. Each user would then need to train the system, by performing grasps for each object occurring in the scenario.

For cases were either potential users are unknown or the grasped objects are not determined in advance, with a recognition rate of about 80%, still an acceptable performance can be achieved for applications where occasional misclassifications are not critical. This might be the case for example public gaming or other entertainment installations.

From the average performance of the examined classifiers a list of classification algorithms that are suited best for the examined problem domain has been compiled. Statistical significance tests suggest that these algorithms perform significantly better than the others in all examined settings. Furthermore, it can be seen that several classifier categories are suited better for this problem domain than others. For example, whereas most function approximators perform reasonably well, Bayesian classifiers in all cases yielded unsatisfactory results. Similarly, most tree (with the exception of LMT and Random Forest) and rule-based classifiers (with the exception of NNge) resulted in comparably low recognition rates when applied to the full problem of distinguishing between all six Schlesinger grasps.

The overall best performing classifier IB1, as indicated by the confusion matrix in Table 6, is most prone to misclassifications between grasps of types palmar and tip, and resp., cylindrical and hook. To a certain extent, these misclassifications can be attributed to an inherently hard separability of these grasp types based on hand shape only (cf. Sammon's Mapping in Section 3.3). However, an examination of the confusion matrices of all classifiers reveals that certain classifiers clearly outperform IB1 and the other classifiers when applied to the specific problem of distinguishing between these hard cases only: For differentiating between tip and palmar grasps, the REPTree and the rule-based ZeroR classifier yielded significantly better recognition rates than all other classifiers. The best separation of palmar and tip graps was achieved by the BayesNet classifier. These observations suggest that a clever combination of classifiers could lead to even higher recognitions rates.

A possibly surprising result was that, while LWL (Local Weighted Learning) did not perform well, the other lazy evaluation algorithms were shown to be the best choice of classifiers in the domain "grasp recognition from raw data glove sensor data".

5 Conclusion

We introduced a systematic approach for the evaluation of classification techniques for recognizing grasps performed with a data glove. In particular, we distinguish between 6 settings that make different assumptions about the user groups and objects to be grasped. A large number of classifiers was compared to draw informed conclusions about achievable recognitions rates in the different settings. Through this, our approach extends previous reports on classifier performance that focused on particular classifiers and settings.

An interesting result of our evaluation is that calibration-free classification of grasps can be performed with reasonable to high reliability for a number of different settings. Surprisingly, algorithms based on lazy learning outperform most of other algorithms in this domain. With respect to the average classification rate over all settings, the IB1 (instance based learner) outperformed all other tested algorithms. This is a suprising result, as lazy learners do not perform any processing of the collected data, which supposedly results in lower generalization ability. Considering the low complexity of implementation, these algorithms seem to be well suited for many VR applications. However, in applications with extreme realtime demands, e.g. requiring over 1000 classifications per second, faster techniques such as the Multilayer Perceptron are better suited. While the total average classification rate of MLP's in our experiments was only less than 1% worse than that of IB1, it had an approximately 35 times shorter runtime.

As indicated by the data analysis in section 3.3, the grasps of the Schlesinger taxonomy are hard to distinguish based on hand shape alone, e.g. palmar and tip. This makes classification based on uncalibrated sensor data – and, presumably, similarly for joint angles values – a challenging task. For our study, this had the advantage that differences between the various classifiers turned out more clearly in the analysis. To achieve better recognition rates, the inclusion of higher level features such as finger bending indexes or contact points will be considered in future work. Other taxonomies with clearer differences between classes in terms of hand poses might also increase recognition rates.

As our results for evaluating the confusion matrices of classifiers show (see section 4.3), some classifiers perform better at distinguishing specific grasp types than others. The strengths of one classifier could compensate the weaknesses of another, so that their combination into a well-tuned meta classifier might lead to a powerful hybrid classification scheme. In future work an in-depth evaluation of various hybrid and meta-classifiers will be performed, which might result in a classifier that distinguishes well between all grasp categories, even the ones which are more difficult to separate. A combination of IB1 (the best overall classifier) with either the ZeroR rules or the REPTree tree classifier and additionally the BayesNet classifier would be a first recommendation. Additionally a thorough parameter optimization process of the applied base classifiers might yield even higher recognition rates than the ones presented here.

Concluding, we believe that the demonstrated approach of calibration-free data glove operation combined with out-of-the-box application of classifiers from a freely available data mining software package greatly simplifies the usage of data gloves.

Acknowledgements

The research described in this contribution is partially supported by the DFG (Deutsche Forschungsgemeinschaft) in the Virtual Workers project.

References

[1] J. Aleotti and S. Caselli. Grasp Recognition in Virtual Reality for Robot Pregrasp Planning by Demonstration. In *IEEE Intl. Conf. on Intelligent Robotics and Automation*, 2006.

[2] C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, November 1995.

[3] G. Bontempi, M. Birattari, and H. Bersini. Lazy learning: a logical method for supervised learning. In *New Learning Paradigms in Soft Computing*, pages 97–136. Physica-Verlag, Heidelberg, 2002.

[4] C. W. Borst and A. P. Indugula. Realistic Virtual Grasping. In *VR '05: Proceedings of the 2005 IEEE Conference on Virtual Reality*, pages 91–98, 2005.

[5] P. K. Chan and S. J. Stolfo. On the accuracy of meta-learning for scalable data mining. *Journal of Intelligent Information Systems*, 8(1):5–28, 1997.

[6] G. F. Cooper and E. Herskovits. A bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9(4):309–347, 1992.

[7] M. Cutkosky. On Grasp Choice, Grasp Models and the Design of Hands for Manufacturing Tasks. *IEEE Trans. on Robotics and Automation*, 5(3), 1989.

[8] T. G. Dietterich. Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms. *Neural Computation*, 10(7):1895–1924, 1998.

[9] S. Ekvall and D. Kragic. Grasp Recognition for Programming by Demonstration Tasks. In *IEEE International Conference on Robotics and Automation*, 2005.

[10] N. Friedman, D. Geiger, and M. Goldszmidt. Bayesian network classifiers. *Mach. Learn.*, 29(2-3):131–163, 1997.

[11] H. Friedrich, V. Grossmann, M. Ehrenmann, O. Rogalla, R. Zöllner, and R. Dillmann. Towards Cognitive Elementary Operators: Grasp Classification using Neural Network Classifiers. In *IASTED International Conference on Intelligent Systems and Control*, 1999.

[12] S. Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1994.

[13] B. Jung, H. B. Amor, G. Heumer, and M. Weber. From motion capture to action capture: A review of imitation learning techniques and their application to VR-based character animation. In *Proceedings VRST 2006 - Thirteenth ACM Symposium on Virtual Reality Software and Technology*, pages 145–154, 2006.

[14] F. Kahlesz, G. Zachmann, and R. Klein. Visual-Fidelity Dataglove Calibration. In *Computer Graphics International (CGI)*, pages 403–410. IEEE Computer Society Press, 2004.

[15] S. Kaski. Data Exploration Using Self-Organizing Maps. *Acta Polytechnica Scandinavica, Mathematics, Computing and Management in Engineering Series No. 82*, March 1997.

[16] J. Napier. The Prehensile Movements of the Human Hand. *The Journal of Bone and Joint Surgery*, 38b(4):902–913, 1956.

[17] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

[18] J. R. Quinlan, P. J. Compton, K. A. Horn, and L. Lazarus. Inductive knowledge acquisition: a case study. In *Proceedings of the Second Australian Conference on Applications of Expert Systems*, pages 137–156, Boston, MA, USA, 1987. Addison-Wesley Longman Publishing.

[19] D. Richards. Ripple down rules: a technique for acquiring knowledge. In *Decision making support systems: achievements, trends and challenges for the new decade*, pages 207–226. Idea Group Publishing, Hershey, PA, USA, 2002.

[20] G. Schlesinger. Der Mechanische Aufbau der Künstlichen Glieder. In M. Borchardt et al., editors, *Ersatzglieder und Arbeitshilfen für Kriegsbeschädigte und Unfallverletzte*, pages 321–661. Springer-Verlag: Berlin, Germany, 1919.

[21] C. Taylor and R. Schwarz. The Anatomy and Mechanics of the Human Hand. *Artificial Limbs*, 2:22–35, 1955.

[22] I. H. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, San Francisco, 2nd edition, 2005.

[23] D. H. Wolpert and W. G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, April 1997.

[24] G. Zachmann and A. Rettig. Natural and robust interaction in virtual assembly simulation. In *Eighth ISPE International Conference on Concurrent Engineering: Research and Applications (ISPE/CE2001)*, pages 425–434, 2001.