GRaD: Gesture Recognition and Disambiguation Framework for Unconstrained, Real-Life Scenarios

Saša Bodiroža
Institut für Informatik
Humboldt-Universität zu Berlin
Berlin, Germany
Email: bodiroza@informatik.hu-berlin.de

Verena V. Hafner
Institut für Informatik
Humboldt-Universität zu Berlin
Berlin, Germany
Email: hafner@informatik.hu-berlin.de

Abstract—**Automatic gesture recognition, used in human-robot and human-computer interaction, has been an active area of research for at least three decades, but there are still recurring issues that hinder its application in unconstrained, real-life scenarios. Interpersonal differences in gesture performance present a prominent issue, which limits the gesture recognition process. Furthermore, these differences are in some cases irrelevant with regard to the message to be conveyed. This work presents a framework for gesture recognition and disambiguation. The goal of this framework is to make the mechanisms of gesture recognition itself as abstract as possible, thus lowering the effect of interpersonal differences, in order to be able to use a simple recognition algorithm. Features that are ignored during the recognition can be used in the disambiguation step, in case they are relevant for this particular message.**

I. Introduction

Gestures are a prominent modality in human-human interaction, usually accompanying speech. Research on gesture recognition methods is ongoing for at least three decades. For example, Bolt presented his work on a speech and pointing controlled interface [1]. However, most of the algorithms work well under certain assumptions, such as the localisation of the person in the scene or the direction of the gesture movement. This results in gesture recognition which performs well under certain conditions, but fails when the assumptions are violated. The goal of this work is to reduce the number of assumptions, in order to have a robust gesture recognition process.

Inter- and intrapersonal variations in gesturing is a recurring issue in gesture recognition. Previous work, which showed the possible complexity of gesture vocabularies and the issues in their design, presented which gestures people associate with particular actions [2], [3]. Current approaches usually impose a limitation on the way how a gesture is performed, for example, by enforcing a particular location in the gestural space or direction of the gesture (e.g. see example gestures in work by Schlömer et al. [4]). In some cases, these differences are not relevant for the exact associated meaning of the gesture. For example, a person might want to make a "circle" gesture. In this case, it should not be relevant whether the circle is large or small, or if the gesture was performed clockwise or counter-clockwise. Further issue is that interacting agents are not always standing still during gesturing in real-life conditions. The framework presented here enables recognition of these type of gestures with a low number of training samples (e.g. with only one gesture per class). However, these abstracted features are highly relevant in some cases. For example, if a user performs a "right to left swipe" gesture, the direction of the motion is important, however it was discarded during the recognition step. To resolve this issue, each gesture class is compared to others to infer if certain gesture pairs are ambiguous, in which case the disambiguation process is performed.

There were efforts to make the gesture recognition more robust [5]. This work presents a concept of a comprehensive framework that aims to alleviate the problems typically encountered in the field of gesture recognition, summarized in Table I. The main aim is recognition of dynamic arm gestures, while static gestures could be recognized to a certain extent, due to their different nature. This is explained in more detail in Section I-B.

Section I-A reviews three gesture taxonomies present in human-robot and human-computer interaction. Section I-B presents an overview of gesture recognition methods and gesture representations and provides use-case scenarios for the proposed framework taking in consideration gesture types. Section II presents the methods used in the framework. Section III presents the framework validation, including the required future validation. Section IV presents a discussion on the contributions of the presented work.

A. Gesture Taxonomies

This section presents taxonomies described by Quek [6], Pavlovic et al. [7], Kendon [8], and Nehaniv et al. [9].

Quek separated gestures into *manipulative* and *communicative* gestures, with regard to their use in human-computer interaction [6]. The purpose of manipulative gestures is object manipulation while communicative gestures are used to transfer information. One qualitative difference is the possibility of visual interpretation. Communicative gestures are visible, providing enough information for their recognition. This means that occluded body parts carry no information required for interpretation. On the other hand, manipulative gestures, e.g. rotation movements of arms while opening a valve, are usually not used for information transfer and, therefore, they might not be fully visible and even interpretable. Communicative gestures can be further separated into *symbols* and *acts*. Symbols are arbitrary in nature. They can be *referential* or *modalizing*. Referential symbols refer to an object or a concept (e.g. holding the index and the middle fingers up as a peace sign).

Modalizing symbol gestures is influencing how a statement is understood (e.g. saying “Have you seen its size?” with the hands wide apart to indicate that the object was large). Act gestures can be *mimetic* and *deictic*. Mimetic gestures imitate the referred object or its use, such as holding both clenched fists together in front of the body and then pulling them closer, as if they are pulling something with a rope. Deictic gestures are gestures that point to the referent. They can be further classified as *specific* (pointing to a particular object), *generic* (pointing to an object which symbolizes the whole class) and *metonymic* (pointing to an object to signify some other entity related to it).

Pavlovic et al. modified the presented taxonomy proposed by Quek to include the unintentional hand movements, such as rubbing hair with the hand, that do not carry any useful information [7].

Kendon presents a narrower definition of a gesture as a movement of body, as already stated above, but with a clear goal of communicating a thought or emotional state [8]. He specifies that gestures are *autonomous*, which can appear without speech, or *gesticulation*, appearing together with speech. Finally, he states the temporal organization of a gesture, so called *gesture phrase*, which consists of a *preparatory movement*, a *nucleus*, which is the informative part of a gesture, and a *return movement*, which returns the hand to the resting position or to the position for a following gesture.

Nehaniv et al. present a gesture taxonomy for application of gestures in human-robot interaction, including *irrelevant* or *manipulative* gestures, *side effect of expressive behavior*, *symbolic gestures*, *interactional gestures* and *referential gestures* [9]. Irrelevant and manipulative gestures correspond to unintentional hand movements and manipulative gestures, as used by Pavlovic et al [7]. Side effects of expressive behavior represent gestures that can be observed during speech, with no particular information transfer role. Symbols are defined in a similar way as they are defined by Quek [6]. Interactional gestures are used to mediate the interaction between partners. Referential gestures are similar to deictic gestures, as defined by Quek [6].

The goal of *manipulative* gestures is usually not information transfer, therefore the focus of this work is on *communicative* gestures, according to the taxonomy presented by Quek [6]. Considering classification by Nehaniv et al. [9], the focus of the work is on symbolic and interactional gestures. Furthermore, the focus is on dynamic uni- or bi-handed gestures, and partially on static gestures, by including dynamic information contained in the preparatory and return movements.

The presented framework covers a part of the communicative gestures from the taxonomy presented by Quek. However, the hand pose provides relevant information for some gestures (e.g. the mentioned peace sign, or a dynamic gesture with a hand posture important for recognition). The framework in its current state does not consider these gestures, but it could be extended to include a recognition of hand poses.

Deictic gestures represent a special case. The primary goal of deictic gestures is to manipulate the attention of an interacting partner to an object or an event of interest [10]. The proposed framework does not handle pointing gestures. However, they have a characteristic signature, containing of the preparatory movement, the hold, during which the finger is still and pointing to a location of interest, and the retraction, taking the hand to the resting position, or preparing for the next gesture. The framework could be extended to include additional branch for processing of pointing gestures, in case they are detected based on their signature.

B. Gesture Recognition and Representations, Application and Advantages of the Proposed Framework

Reviews of gesture recognition approaches can be found in [11] and [12] and examples include hidden Markov models, dynamic time warping, finite state machines and time-delayed neural networks for dynamic gestures. Gestures can have different representations – absolute or relative 2D or 3D trajectories or first derivative of the trajectories common in recognition with dynamic time warping [13], [14], [15], quantizing the trajectory for processing by a HMM [16], using orientation histograms [17], using a fixed-length histogram of features for detection using a SVM [18] and so on.

Recently, Suarez and Murphy [19] published a review of state-of-the-art methods in hand gesture recognition and identified some of the still-present issues in gesture recognition. One of the issues they mention is the location and orientation of a person, such as when they are not actively facing the sensor. Another issue is unfamiliarity of the person with a gesture set used by the system. This only stresses out the importance of increasing the tolerance of the gesture recognition algorithm to both inter- and intra-personal variances. The methods outlined in this work alleviate to some extent these issues. Alternatively, due to the low number of training gestures required, a particular gesture could be easily retrained if needed.

Wachs et al. [20] analysed possible applications of hand gestures in human-computer interfaces. Some of the identified costs and benefits of using hand gestures are: price, responsiveness, user adaptability and feedback, learnability and low mental load, accuracy (detection, tracking and recognition), interaction space. The presented framework was partially implemented and tested. It is *responsive*, running in real time, due to the relatively simple algorithms which are used. The framework provides partial *user adaptability*. Because of the preprocessing, as explained in the Section II-B, the recognition algorithm can be kept simple and it can work reliably with a low number of training samples. Furthermore, the preprocessing of the gesture trajectories reduces *mental load* and improves *learnability*, as some variations in performances of the same gesture are tolerated. *Interaction space* is not hindered, and the user can freely move around while gesturing.

When people gesture, they tend to perform some gesture classes in various ways, but still resulting in the same gesture. Some of the features causing the difference are usually the size of the gesture, its location in the gestural space, the speed of the hand(s) that perform the gesture and direction of the hand(s).

These differences can carry meaningful information, but sometimes they are irrelevant. For example, a person might want to indicate that an object had a circular shape and use a mimetic gesture that represents a circle. There are different possible ways to perform this gesture, which can be observed

TABLE I. ISSUES IN GESTURE RECOGNITION

Issues	Proposed solutions
Interpersonal orientation	Frame of reference transformation
Gestural size	Normalization, standardization
Gestural location	Trajectory alignment (first points or centroid)
Varying distribution of trajectory samples	Uniform resampling
Varying starting and ending points of a gesture	Bag of points

– it can be performed with the hand moving clockwise or counter-clockwise, or the resulting circle can vary in size between different persons.

In other cases, these differences are relevant, and should be used to disambiguate between particular gestures, e.g. whether the round object was big or small, which could be indicated with the size of the "circle" gesture.

Table I summarizes issues in gesture recognition and proposed solutions presented in this work.

II. GRaD FRAMEWORK

The goal of the framework for gesture recognition and disambiguation (GRaD) is to handle the issues that are common in gesture recognition in real-life scenarios.

The idea driving the development of this framework was that:

- gestures, regardless of whether the mentioned features are relevant or irrelevant, can be recognized using low number of training samples, and
- disambiguation, performed after recognition, discerns different gestural classes based on training, such as "large circle" or "small circle" in case where of two groups of training samples, one related to the former, and one to the latter class, or just a "circle" if there was only one class for general gesture of "circle".

A low number of training samples is a highly important feature of the framework, enabling gesture recognition to become an easy to use, ubiquitous interface, having in mind that in adaptive systems, new gestures need to be added and current ones need to be modified while the system is in use.

The procedure consists of training and recognition. This framework assumes the use of supervised training, that is labelling each training trajectory with the assigned meaning. The framework is designed to be used with 2D or 3D trajectories. Depth sensors have recently became affordable and obtaining gesture trajectories. However, the data they provide is usually noisy and contains outliers. In addition, some features are not relevant for the recognition itself, while being relevant for disambiguation in some cases. In order to have robust gesture recognition, raw data should be preprocessed to smooth the recorded trajectory and to remove particular features, such as gestural size, direction, velocity and velocity profile, location in the gestural space and starting and ending points of the gesture, that are not relevant for the recognition of the basic gesture. The idea is to make the gesture recognition invariant with regard to these features. Furthermore the recognition is performed on a trajectory which is standardized and aligned with the comparison gesture.

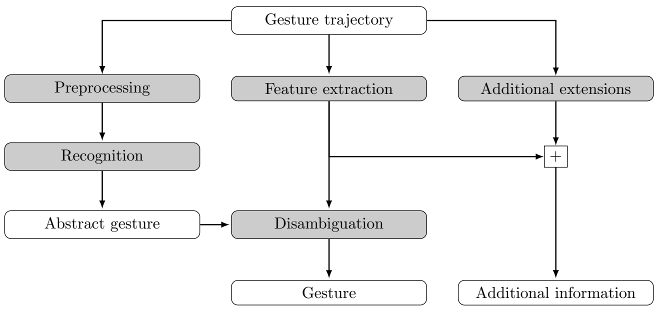

Fig. 1. Proposed Gesture Recognition and Disambiguation (GRaD) framework

Figure 1 represents the general outline of the framework. A gesture trajectory is obtained through a sensor, e.g. a depth camera. The trajectory is preprocessed before the recognition. The output of the recognition algorithm is the identifier of the recognized gesture. In case there is some ambiguity, i.e. if this identifier is mapped to more than one gesture, then the identifier together with additional features is provided to the disambiguation algorithm to determine what is the observed gesture. Future work might also include gestural plane extraction, as some of the 3D gestures are performed in a 2D plane [21].

A. Data acquisition

A gesture is represented as a time sequence of 3D joint positions. Commonly, these joints are some or all of the arm joints, but depending on the use case, it could also include other body joints. Joint positions can be obtained using different sensors, e.g. with RGB cameras, motion trackers or accelerometers. In recent years affordable depth sensors, such as ASUS Xtion or Microsoft Kinect, are gaining more interest, because they can provide 3D coordinates of body joints with a $30Hz$ frame rate, which is satisfactory for everyday application in human-robot and human-computer interaction.

B. Data preprocessing

Data preprocessing consists of five stages and includes the following steps: frame of reference transformation, normalization or standardization, trajectory alignment, uniform resampling and direction-invariance. Throughout the rest of the work, the term "abstract gesture" will represent a gesture trajectory with discarded features. For example, the "swipe left" gesture, with the hand moving from right to left in front of the body, and the analogous "swipe right" gesture would be recognized as the same abstract gesture, that is they would be represented with a very similar direction-invariant trajectory.

1) Frame of reference transformation: Some gesture recognition algorithms tend to fail when the interacting partners are not standing still during gesturing. The reason is that the data provided by a depth sensor is grounded in the sensor's frame of reference. Therefore, the first step is to ground the points of the observed trajectory, $\mathbf{A_i}$ to an anchor point fixed on a person, $\mathbf{A_{anchor}}$, e.g. to the point between the hip joints. Doing this, trajectories are observed in the person's frame of reference. However, this is still sensitive to the rotation of the person, relative to the axis drawn through the person's and the sensor's frames of reference. To resolve this issue, the absolute orientation of the anchor point is obtained and the difference vector obtained in the first step is multiplied by the rotation matrix, $\mathbf{R}$, providing the person's position- and rotation-invariant gesture representation, as presented by Bodiroža et al. [15]. Similar approach was presented by Chaaraoui et al. [22].

$$\mathbf{A_{i,invariant}} = (\mathbf{A_i} - \mathbf{A_{anchor}})\mathbf{R} \qquad (1)$$

2) Normalization and standardization: To enable gesture recognition with low number of training samples, all observed and trained gestures are either normalized or standardized. Normalization scales the trajectory to a fixed range, e.g. $[0 \ldots 1]$, while standardization scales the trajectory with regard to the mean and standard deviation, so that the new trajectory has zero mean and unit variance (i.e. $\mu = 0, \sigma = 1$).

Normalization can be performed with independent or dependent dimensions. The independent approach will scale the trajectory along all three dimensions to the same, fixed range, regardless of the domain ratio between each pair of dimensions. However, if a gesture is most prominent in only one or two dimensions, this will result in a reduced recognition rate, due to the fact that initially not all dimensions carry the same weight, but after rescaling they will all have contribute with a similar weight during recognition, as the low-contributing dimension will provide similar amount of information.

Prior to normalization, the data are smoothed using Gaussian filter. The smoothing is performed to remove outliers, which are commonly present in data acquisition. The algorithm outline is presented in Algorithm 1.

Algorithm 1 Data smoothing and normalization

```
Require: An array of n-dimensional points $A$.

 1: function SMOOTHNORMALIZE($A$)
 2:     $\sigma \leftarrow 1$
 3:     $A \leftarrow Gauss1DSmooth(A, \sigma)$
 4:     for all Dimension d in A do
 5:         $max_d \leftarrow FindMaxValue(d_A)$
 6:         $min_d \leftarrow FindMinValue(d_A)$
 7:         $diff_d \leftarrow max_d - min_d$
 8:     $PromDim \leftarrow Index(Max(diff_{d1}, \ldots, diff_{dn}))$
 9:     for all Dimension d in A do
10:         for all Point a in A do
11:             $a_d \leftarrow (a_d - min_d)/diff_d$
12:     for all Dimension d in A except PromDim do
13:         $scale_d \leftarrow diff_d/diff_{PromDim}$
14:         for all Point a in A do
15:             $a_d \leftarrow a_d \cdot scale_d$
16:     return $A$
```

Standardization presents a similar process, by first finding mean and standard deviation for each dimension of the sequence $\mathbf{A}$, and subtracting the mean from each dimension of each point and dividing by the standard deviation.

3) Trajectory alignment: In order to make the recognition invariant with regard to the location where the gesture is performed within the gestural space, the observed sequence $\mathbf{A}$ needs to be aligned with the trained sequence $\mathbf{B}$, to which the gesture is compared.

This alignment can be done in various ways. One way is to align the two sequences $\mathbf{A}$ and $\mathbf{B}$ with regard to their starting points A_1 and B_1, i.e., to compute the difference vector between their starting points A_1 and B_1 and subtract this vector from all subsequent points of sequence $\mathbf{A}$, as presented in Algorithm 2 [15].

Algorithm 2 Trajectory alignment using the difference between first points

```
Require: $A$ is an observed and $B$ is a trained sequence.

1: function DIFFALIGNMENT($A, B$)
2:     $d_{x,y,z} \leftarrow b_{1x,y,z} - a_{1x,y,z}$
3:     for all Point a in A do
4:         $a_{x,y,z} \leftarrow a_{x,y,z} - d_{x,y,z}$
5:     return $A$
```

Another approach is to find the centroid of sequence $\mathbf{B}$ and subtract it from sequence $\mathbf{A}$, as presented in Algorithm 3.

Algorithm 3 Trajectory alignment using the centroid

```
Require: $A$ is an observed and $B$ is a trained sequence.

1: function CENTROIDALIGNMENT($A, B$)
2:     $sum_x, sum_y, sum_z \leftarrow 0$
3:     for all Point b in B do
4:         $sum_{x,y,z} \leftarrow sum_{x,y,z} + b_{x,y,z}$
5:     $CoG_{x,y,z} \leftarrow sum_{x,y,z}/len(B)$
6:     for all Point a in A do
7:         $a_{x,y,z} \leftarrow a_{x,y,z} - CoG_{x,y,z}$
8:     return $A$
```

Testing showed that the recognition was in general better when using the centroid, especially taking in consideration when it is combined with the bag-of-points method, explained in Section II-B5, to make the recognition direction-invariant.

4) Uniform resampling: The next step is re sampling of the trajectory to obtain one with a uniformly-distanced points. This is performed in order to diminish the effect of velocity profiles of different persons, as well as the difference in distribution of points when the same gesture (e.g. the "circle" gesture) has different starting points (e.g. beginning the "circle" gesture in the upper or the lower section). If this is not performed, due to these differences the observed and trained gestures could not always be well aligned and matched

5) Direction invariance: Final preprocessing step is performed in order to enable recognition of a specific case of gestures, which can be performed in different directions. For example, the "circle" gesture can be performed clockwise or counter-clockwise, and the "hand wave" could be performed starting from left to right, or from right to left. A bag-of-points approach is used for making the recognition direction-invariant. Both the observed gesture and the one to which it is compared are represented using a bag-of-points model.

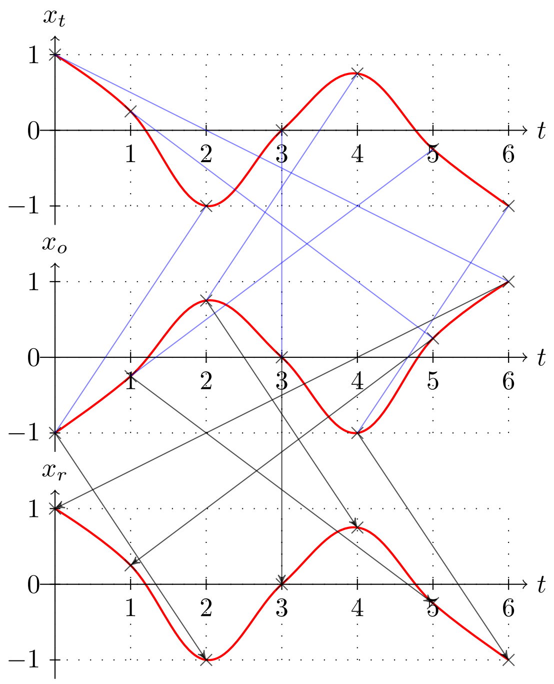

Fig. 2. Direction invariance: remapping of points of the observed sequence (second plot) by comparing them to the trained sequence (first plot) in order to obtain the sequence (third plot) that best matches the trained one. The plots present only one dimension in time for illustration purposes. The observed sequence is the trained sequence performed in the other direction.

A bag-of-points is a concept analogous to a bag-of-words in natural language processing. A gesture trajectory is stored in a multiset, an unordered sequence of points, where one point in space is allowed to have multiple instances. Each point from the observed gesture is matched to every point in the trained gesture in order to find the closest point. Distance between two points is determined using Euclidean distance as a measure. A "maximum mapped points" parameter defines what is the maximum number of points from one gesture that can be mapped to one point of another gesture. This parameter prevents the case of most of the points from one gesture being mapped into only one or two points of the other gesture. Figure 2 illustrates the described procedure.

C. Gesture recognition

Due to the preprocessing of the gesture trajectory, the gesture recognition algorithm does not require many training samples in order to have acceptable recognition rates. The algorithm was tested on example gestures, namely "frontal circle", by drawing a circle along the frontal plane, "transverse circle", by drawing a circle along the transverse plane, "wave", by performing a hand wave, right-left-right motion along the frontal plane (note that the trajectory of this gesture overlaps with the "frontal circle" trajectory) and a "stop" gesture, by extending the arm in front of the body, holding it for 1-2 seconds, followed by the return movement. Training was performed with one sample per gesture class, performed by one person, while two person were gesturing during testing.

The framework was able to discern those mentioned gestures, regardless of the direction, starting point, size, speed or location. It is worth noting that it was also successful in the recognition of "frontal circle" and "wave" gestures. When the testing was performed without preprocessing, the algorithm was only able to recognize gestures if they were performed only in the fashion of the training samples.

This shows that the framework was successful in alleviating intra- and interpersonal differences in gesturing, therefore making the recognition more robust and suitable for use in unconstrained, real-life scenarios [23].

D. Gesture disambiguation

During the data preprocessing stage, certain features were discarded. This leads to induced ambiguity between particular subsets of gestures, which are disambiguated with these features. An example of this problem are the gestures "swipe left" and "swipe right". Since gestural direction is one of the features that is discarded during the preprocessing, these two gestures can be misclassified, resulting in false positives.

The solution is to apply machine learning techniques, to automatically learn when these features are relevant. Supervised (e.g. feed-forward neural networks) or unsupervised (e.g. self-organizing maps) methods could be applied to perform classification or clustering of the observed gesture, respectively, taking in consideration both the output of the gesture recognition algorithm and the features that were removed during the preprocessing.

1) Feature extraction: Features that were discarded for the purpose of gesture recognition are extracted from the input gesture trajectory, whereby it is suggested to extract these features after initial smoothing.

- Gesture size is represented with the difference of maxima and minima along the three dimensions of the trajectory.
- Gesture location in the gestural space is represented as the centroid of the gesture trajectory relative to an anchor point, e.g. the point between the hip joints.
- Beginning and end of the gesture trajectory are represented as those points relative to the anchor point.
- Gesture direction is represented as the vector along two most prominent dimensions, as the difference between the first occurring extremum and the second occurring extremum along a dimension.
- Simplified gesture velocity as a time duration of the particular gesture.

2) Disambiguation: To evaluate if some pairs of gesture classes need to be disambiguated, the abstract gesture trajectories are compared for every 2-combination from a set of gesture classes. If the difference between two trajectories is below an empirically-established threshold, then disambiguation needs to be performed. Otherwise, if the difference between a particular abstract gesture trajectory and every other trajectory is above the threshold, then the output of the recognition algorithm is sufficient.

Linear classifiers can be used for the disambiguation. Given the abstract gesture identifier, provided by the recognition algorithm, together with the features that are extracted from the trajectory, and the expected output, in the form of a gesture identifier, a classifier, such as a support vector machine, is trained to disambiguate between two gesture classes.

III. Framework validation

The aim of the framework is gesture recognition robust to the intra- and interpersonal differences of gestures, such as size, location and speed of a gesture and the orientation and position of the person relative to the sensor or to the robot. Parts of the presented work have already been demonstrated and evaluated [15], [23], [24]. However, gesture disambiguation requires future validation.

Furthermore, due to the preprocessing and the abstraction it provides to the gesture recognition algorithm, a dataset, containing gesture performances varying in size, speed, location and direction within the same class both with the person being static and moving around, should be specifically designed in order to fully test the effects of the proposed framework.

IV. Discussion and Conclusion

This work presented a framework for gesture recognition and disambiguation. The main contribution of this work is that the framework provides robust recognition of dynamic and to certain extent static arm gestures, by discarding certain features of the gesture during the recognition and employing that information in the disambiguation step, following the recognition, if it is automatically evaluated that those features were relevant for a particular combination of gesture classes.

By making the gesture as abstract as possible, i.e. observing the gesture as an abstract gesture trajectory, it is possible to keep the algorithm very simple, for example using dynamic time warping [15], [25], [14], [13], therefore requiring a low number of training samples. Furthermore, a simple linear classifier could be employed for the disambiguation. This enables the robust recognition in real-life scenarios by lowering the requirements on the user side, such as the number of training samples per gesture class, and by lowering the complexity of the implementation.

The focus of future work should be on extending the framework to include the recognition of static gestures and gestures where the hand pose is relevant.

Acknowledgment

The present work has been conducted as part of the EARS (Embodied Audition for RobotS) Project. The research leading to these results has received funding from the European Union's Seventh Framework Programme (FP7/2007-2013) under grant agreement number 609465.

References

[1] R. A. Bolt, ""put-that-there": Voice and gesture at the graphics interface," in *Proceedings of the 7th Annual Conference on Computer Graphics and Interactive Techniques*. New York, NY, USA: ACM, 1980, pp. 262–270.

[2] S. Bodiroža, H. I. Stern, and Y. Edan, "Dynamic gesture vocabulary design for intuitive human-robot dialog," in *Proc. 7th ACM/IEEE Int. Conf. on Human-Robot Interaction*. New York, NY, USA: ACM, 2012, pp. 111–112.

[3] H. Stern, J. Wachs, and Y. Edan, "Human factors for design of hand gesture human - machine interaction," in *IEEE International Conference on Systems, Man and Cybernetics*, vol. 5, Oct 2006, pp. 4052–4056.

[4] T. Schlömer, B. Poppinga, N. Henze, and S. Boll, "Gesture recognition with a wii controller," in *Proceedings of the 2Nd International Conference on Tangible and Embedded Interaction*. New York, NY, USA: ACM, 2008, pp. 11–14.

[5] A. Sadeghipour and S. Kopp, "A probabilistic model of motor resonance for embodied gesture perception," in *Intelligent Virtual Agents*, ser. Lecture Notes in Computer Science, Z. Ruttkay, M. Kipp, A. Nijholt, and H. H. Vilhjálmsson, Eds. Springer Berlin Heidelberg, 2009, vol. 5773, pp. 90–103.

[6] F. K. Quek, "Eyes in the interface," *Image and Vision Computing*, vol. 13, no. 6, pp. 511 – 525, 1995.

[7] V. I. Pavlovic, R. Sharma, and T. S. Huang, "Visual interpretation of hand gestures for human-computer interaction: A review," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 19, pp. 677–695, 1997.

[8] A. Kendon, "Current issues in the study of gesture," in *The Biological Foundations of Gestures: Motor and Semiotic Aspects*, J.-L. Nespoulous, P. Perron, and A. R. Lecours, Eds. Lawrence Erlbaum Associates, 1986, pp. 23–47.

[9] C. Nehaniv, K. Dautenhahn, J. Kubacki, M. Haegele, C. Parlitz, and R. Alami, "A methodological approach relating the classification of gesture to identification of human intent in the context of human-robot interaction," in *IEEE International Workshop on Robot and Human Interactive Communication, RO-MAN*, 2005, pp. 371–377.

[10] N. Masataka, "From index-finger extension to index-finger pointing: Ontogenesis of pointing in preverbal infants," in *Pointing: Where language, culture, and cognition meet*, S. Kita, Ed. Mahwah, New Jersey, USA: Lawrence Erlbaum Associates, 2003, pp. 69–84.

[11] S. Rautaray and A. Agrawal, "Vision based hand gesture recognition for human computer interaction: a survey," *Artificial Intelligence Review*, pp. 1–54, 2012.

[12] S. Mitra and T. Acharya, "Gesture recognition: A survey," *IEEE Trans. Syst. Man Cybern. C, Appl. Rev.*, vol. 37, no. 3, pp. 311–324, May 2007.

[13] A. Corradini, "Dynamic time warping for off-line recognition of a small gesture vocabulary," in *Proc. IEEE ICCV Workshop on Recognition, Anal., and Tracking of Faces and Gestures in Real-Time Syst.*, 2001, pp. 82 –89.

[14] G. A. ten Holt, M. J. T. Reinders, and E. A. Hendriks, "Multi-dimensional dynamic time warping for gesture recognition," in *Proc. of the Conf. of the Advanced School for Computing and Imaging*, 2007.

[15] S. Bodiroža, G. Doisy, and V. V. Hafner, "Position-invariant, real-time gesture recognition based on dynamic time warping," in *Proc. 8th ACM/IEEE Int. Conf. on Human-Robot Interaction*. Piscataway, NJ, USA: IEEE Press, 2013, pp. 87–88.

[16] J. Yamato, J. Ohya, and K. Ishii, "Recognizing human action in time-sequential images using hidden markov model," in *Computer Vision and Pattern Recognition, 1992. Proceedings CVPR '92., 1992 IEEE Computer Society Conference on*, Jun 1992, pp. 379–385.

[17] W. T. Freeman and M. Roth, "Orientation histograms for hand gesture recognition," in *International Workshop on Automatic Face and Gesture Recognition*, vol. 12, 1995, pp. 296–301.

[18] N. Dardas, Q. Chen, N. D. Georganas, and E. Petriu, "Hand gesture recognition using bag-of-features and multi-class support vector machine," in *Haptic Audio-Visual Environments and Games (HAVE), 2010 IEEE International Symposium on*, Oct 2010, pp. 1–5.

[19] J. Suarez and R. Murphy, "Hand gesture recognition with depth images: A review," in *IEEE RO-MAN*, Sept 2012, pp. 411–417.

[20] J. P. Wachs, M. Kölsch, H. Stern, and Y. Edan, "Vision-based hand-gesture applications," *Commun. ACM*, vol. 54, no. 2, pp. 60–71, Feb. 2011.

[21] S. Berman and H. Stern, "Sensors for gesture recognition systems," *IEEE Trans. Syst. Man Cybern. C, Appl. Rev.*, vol. 42, no. 3, pp. 277–290, May 2012.

[22] A. A. Chaaraoui, J. R. Padilla-López, P. Climent-Pérez, and F. Flórez-Revuelta, "Evolutionary joint selection to improve human action recognition with rgb-d devices," *Expert Systems with Applications*, vol. 41, no. 3, pp. 786 – 794, 2014.

[23] G. Doisy, A. Jevtić, and S. Bodiroža, “Spatially unconstrained, gesture-based human-robot interaction,” in *Proc. 8th ACM/IEEE Int. Conf. on Human-Robot Interaction*. Piscataway, NJ, USA: IEEE Press, 2013, pp. 117–118.

[24] A. Jevtić, G. Doisy, S. Bodiroža, Y. Edan, and V. V. Hafner, “Human-robot interaction through 3d vision and force control,” in *Proc. 9th ACM/IEEE Int. Conf. on Human-Robot Interaction*. New York, NY, USA: ACM, 2014, pp. 102–102.

[25] M. A. Bautista, A. Hernández-Vela, V. Ponce, X. Perez-Sala, X. Baró, O. Pujol, C. Angulo, and S. Escalera, “Probability-based dynamic time warping for gesture recognition on RGB-D data,” in *Advances in Depth Image Analysis and Applications*, ser. Lecture Notes in Computer Science, X. Jiang, O. Bellon, D. Goldgof, and T. Oishi, Eds. Springer Berlin Heidelberg, 2013, vol. 7854, pp. 126–135.