Human Robot Interaction using Dynamic Hand Gestures

Zuhair Zafar[1], Daniel A. Salazar*[1], Salah Al-Darraji[2], Djordje Urukalo[3], Karsten Berns[1], and Aleksandar Rodić[3]

[1] University of Kaiserslautern, Department of Computer Science, RRLAB, 67663 Kaiserslautern, Germany
{zafar; berns}@cs.uni-kl.de, daniel98mex@gmail.com*
http://rrlab.cs.un-kl.de
[2] University of Basrah, Department of Computer Science, Basrah, Iraq
aldarraji@uobasrah.edu.iq
[3] University of Belgrade, Mihajlo Pupin Institute, Robotics Laboratory, 11060 Belgrade, Serbia
{djordje.urukalo; aleksandar.rodic}@pupin.rs

Abstract. This paper describes the implementation of a robust dynamic hand gesture recognizer using a depth sensor. The recognizer uses only depth image information, and the hand position provided by a hand tracker library, in order to construct its feature vectors. The recognizer builds two types of feature vectors to increase accuracy; the frame feature vectors that describe a static hand, and the sequence feature vectors that describe a contiguous segment of frames. The recognizer also uses two statistical classifiers. The frame feature vectors are utilized by the frame classifier. The results of the classifier, then become part of the sequence feature vector, which in turn are utilized by the sequence classifier. The results show that the accuracy of the recognizer increases more than twice, when using both classifiers. The recognizer also does not make any assumption for when a gesture begins or when it ends. Instead it learns to differentiate between noise, and a real gesture. A humanoid robot, ROBIN, is used for validation of the approach for human-robot interaction.

Keywords: Dynamic hand gestures, Human robot interaction, Voxel mapping, Classification, Feature vector

1 Introduction

Human gesture recognition is an important research area due to its utility in business, and a person's daily life. The aim is to create a system that interact with human beings efficiently and in a more natural manner. Gestures play an important role in the communication process; by enhancing the verbal information content and in some cases by completely substituting it. Gestures can be categorized as emblems or as illustrators. The former category is for speech-independent gestures, while the later is for speech-dependent ones [2]. Emblems

have a meaning by themselves and can be usually described with a few words. It is possible however that the emblems meaning can be modified with speech [4]. The use of a particular emblem also depends on the speaker and the audience; some emblems are mainly used for children, or appropriate in specific situations. The total number of emblems depends on the person and his culture. For example, the number of emblems used by students varies from less than 100 for the United States, to more than 250 for Israel [4]. Illustrators do not have a concrete meaning, but they can enhance the meaning of speech, and help the audience understand and remember its message. Some uses of illustrators are: to mark punctuation, give emphasis, refer to an object, indicate direction, and convey the shape of an object, its movement or its size. Hand gesture recognizers are usually divided into static and dynamic. The former includes emblems such as the thumbs up, the Okay sign and the peace sign. The latter include gestures such as waving, saying no, come here and scolding. Static hand gestures are easier to recognize since the recognizer just has to recognize a particular geometrical shape without worrying about timing. Dynamic hand gestures on the other hand, sometimes have to make assumptions for when a gesture starts and when it ends. For this reason it can become very resource intensive, if the algorithm tries many different options in real time, or completely miss a gesture, if the assumptions are incorrect.

One common algorithm used for dynamic hand gesture recognition, is Dynamic time warping (DTW). This algorithm works by minimizing the cost between two mapped time sequences: the sequential hand/body joint states for an *ideal* pre-recorded gesture denoted as a template, and the desired input pattern. The final mapping is a set of ordered pairs that links the time frames of one sequence to the time frames of the other sequence. When researchers use DTW, they sometimes use body/hand joint positions as feature vectors [1][7]. As pointed out in [1], not all joint positions are equally important for a given gesture, thus a weighted DTW is preferable [1]. Nonetheless, joint positions can be hard to obtain without special hardware, due to occlusion, perspective, segmentation, and recognition issues. For this reason, it could be necessary to accommodate other kind of feature vectors.

Another common algorithm used for dynamic hand gesture recognition is Hidden Markov Models (HMM). It is possible that this algorithm is able to recognize hand gestures continuously, without having a start or end time frame. This can be possible if the HMM's topology is fully connected and includes a noise output state. A disadvantage of the HMM's learning algorithm is that it requires large training sets [1]. This is specially the case as the number of transitions increases quadratically for this topology. Other topologies can be used that use less transitions, but the algorithm can become less flexible, more complex, and the researcher must make creative assumptions as to how a gesture starts and how it ends.

Some other algorithms, used for dynamic hand gestures recognition include: Deep belief networks [6], Support Vector Machines (SVM) [10], Finite State Machines (FSM) [5], and k-nearest neighbours (k-NN) [9]. These algorithms,

generally need a starting and ending time for each gesture, or each video must contain one gesture. In addition, some algorithms require that the person who performs the gesture must maintain a certain distance from the camera, needs to perform the gesture in a certain way, or with a similar hand displacement.

Section 2) talks about proposed approach and major modules of the approach like voxel transformation, generation of feature vectors and classification. Section 3 describes experimentation scenario and evaluation of the system. Finally in section 4, we conclude our paper.

2 Proposed Approach

The process by which the dynamic hand gesture recognizer determines a subject's gesture, is shown in Fig. 1. In total, the recognizer uses two different functions that extract feature vectors from data, and two different classifiers. The recognition process is divided into different functions. Functions are represented as rectangles or squares, while data is represented with an ellipse. At each time step, every function receives a new value and outputs the result. This result will be used by another function, on the next time step. The process receives a depth image as the input, and outputs the corresponding classification for the current gesture.

2.1 Voxel Transformation and Segmentation

For recognition of hand gestures, a depth sensor based camera is used. OpenNI is an open-source software that interacts with various Kinect type sensors, to retrieve visual information. Natural Interaction Middleware (NITE), is a middleware library pack, that uses OpenNI to track human skeleton joint positions. This library can also track the hand and separate users from the background. We use this library to get 3D depth coordinates of the hand. The depth image "x" and "y" coordinates, suffer from the perspective distortion due to the camera's focal length, thus, there is no easy conversion to world coordinates. With perspective distortion, an object appears to be bigger or smaller depending on its relative distance to the camera. This makes the segmentation process, and the creation of meaningful feature vectors harder.

The voxel transform function takes as input a depth image, and generates a bi-dimensional voxel matrix. The reason for this transformation is to relate in a meaningful way the depth axis, with the x-axis and the y-axis. This transformation also lessens the effects of the perspective distortion since each voxel has its own x, y, and z values. The transformation also makes segmentation easier, since a fixed cuboid can be used for segmentation. To keep the hand perception software simple, color (RGB) image is not taken into account. Although, the RGB camera can provide valuable information, this can also make the recognizer more complex. The bi-dimensional voxel matrix is then converted into feature vectors by the "Segmentation + Frame Feature Extraction" function. Segmentation process includes following steps:

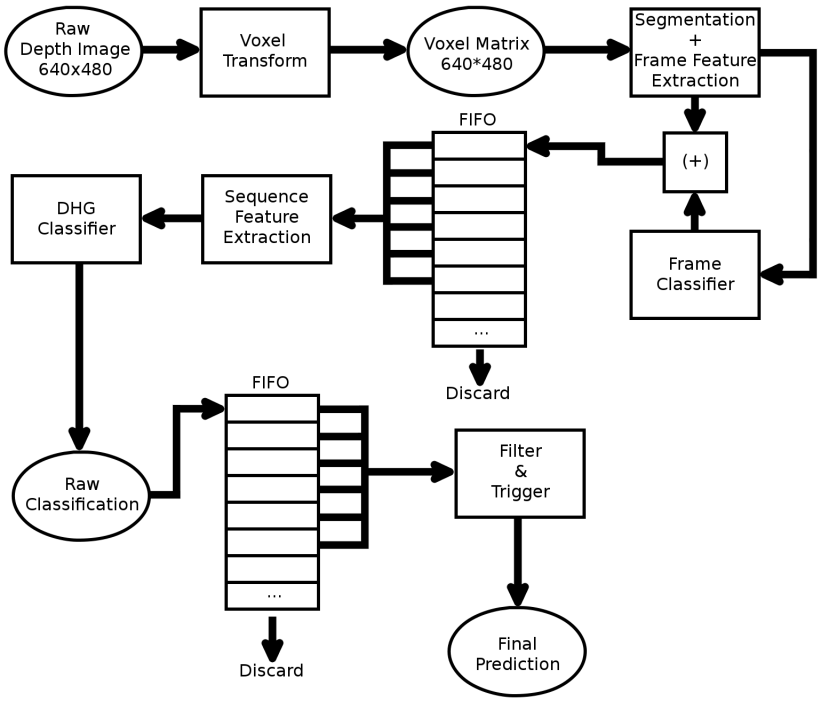

Fig. 1. General process that the proposed dynamic hand gesture recognizer utilizes to classify a gesture. The concatenation function is denoted as "(+)". The First in – first out (FIFO) data structure is only used to indicate how data is stored (Access to its elements can be done at any point). At each time step, a new element is inserted in the FIFO buffer, and, an old element is removed

1. Retrieve the hand 3D center from the NITE library.
2. Find the closest voxel to the hand center and update the hand center position. In case none is found, segmentation fails. This is done in order to prevent tracking system errors from further affecting the recognition software.
3. Trim the voxel array with a 30 cm edge cube, centred at the hand center position.
4. In order to remove nearby objects such as other body parts, remove voxels that are not connected to the hand. This is done by removing voxels that cannot form a path to the hand center voxel. There is an edge between two voxels if their pixel position value differs only by 1 pixel and the depth distance is less than 3 cm.

2.2 Feature Vectors

Useful feature vectors for dynamic hand gestures have to represent important traits such as hand posture and movement (velocity, total displacement, etc). Two steps are performed in order to create a feature vector that trains or tests a machine learning model for the classification task. The first step consists of reducing the information of every frame by building "frame" feature vectors. The second step is to convert a continuous segment of frame features, into the final feature vector, the sequence feature vector.

Frame Feature Vectors Multiple features are used and then combined together to make a frame feature vector. It includes intuitive features like validity of hand and hand's geometric center position. It also includes shape of the hand. This feature represents the square root and normalized eigen values vector and the eigen vectors of the covariance matrix. The eigen values represent a rough model of the hand's 3D geometry. For example, if all eigen values are equal, then the hand appears as a sphere to the recognizer. If the last value is 0, then the hand appears as an ellipse, or a line if the second value is also 0. One important feature is hand size which is computed by square root of the mean square distance of every voxel of the hand to the hand center in 3D world coordinates. Last feature in frame feature vector is based on the hand projections. The feature is made up of 6 projection vectors of the hand, containing 6 components each. This vector is normalized since how big or wide is the hand, has no relevance for the recognizer.

Sequence Feature Vectors The sequence features are build out of 20 continuous frame features that have at least 15 valid frames. Different relevant features are explored. One of the feature used is size of the trajectory relative to the size of the hand. This can be computed by taking the square root of the mean square distance of the hand center 3D to the mean hand center 3D and then divided by the mean hand-size. Furthermore, shape of the trajectory is also computed from the hand center 3D to build a covariance matrix. Another important feature included in feature vector is the mean velocity of the hand. A feature, *pause2center*, is also defined. This feature represents the normalized mean vector, from the hand center 3D to the mean hand center 3D, when the hand is not moving fast enough (pause state). The pause state is detected when the hand speed is slower than a minimum-speed threshold composed of the median velocity and mean hand-size.

Additionally to differentiate clockwise and counter-clockwise motion of hand, *cross2product* feature is included. This feature represents the mean square magnitude cross product vector between two sequential normalized velocity vectors $(\widehat{v_i}, \widehat{v_j})$. The purpose of squaring the magnitude of the cross product is to decrease the effect of velocity vectors that change slowly or do not change in angle.

$$\begin{aligned} v_{cross} = \widehat{v_i} \times \widehat{v_j} \qquad & j = i+1, i < frameCount, \{i, j\} \\ & cross2product = mean(v_{cross}.\|v_{cross}\|) \end{aligned} \tag{1}$$

2.3 Random Forest Classification

A decision tree is a popular method used in machine learning and data mining. Decision trees have nodes with various conditions, and depending on the data of a given sample, the branch that satisfies the nodes condition is taken. The problem with decision trees is their tendency to overfit, as they grow in depth, the number of samples decreases, thus the variance increases. In other words, the statistical significance of a node's population decreases as the number of samples decrease. The random forest algorithm uses the bootstrap aggregation (bagging) technique to reduce variance [3]. This technique generates new training sets for each tree. The training sets are created by sampling the input training set with replacement. Furthermore, each random tree uses a random subset of features to create more variation, and to avoid choosing the same features to create rules every time. Other statistical methods can also be used, but random forest do not need any adjustment and are robust to noise.

Frame features are used to train the first classifier using random forests. The classification is done by only using the shape of the hand, and not its movement/trajectory. The reason for doing this is because the hand shape can be lost when making the sequence feature vectors. This frame level classifier is used in the next stage to generate sequence level features. The sequence feature vector, is then divided into the training, and testing datasets. Afterwards, the sequence feature vectors are utilized to build the final sequence classifier. 15 subjects have participated during the training phase and approximately 250,000 frames have been tagged with start and end frames of the gestures.

3 Experimentation and Evaluation

To test the system, a simple HRI application was built. This application uses a RGB-D camera for its input, and the voice of a humanoid robot and a computer screen for its output. When utilizing this application with humanoid robot, ROBIN [8], the humanoid robot reacts to the gesture of an actor, by indicating the name of the current gesture, or by responding to it. Figure 2 shows different dynamic hand gestures that have been used in this experimentation. Most of these gestures are interactive gestures, for example, expressing *no* with index finger or waving to say hello or making a talking hand or showing the clockwise or counter-clockwise motion of hand.

To measure the accuracy of the proposed system for the hand gesture recognition, 2 different experiments were performed. In the first experiment, actors performed the gestures in front of ROBIN robot, where ROBIN, uses its depth sensor, captures the hand and recognizes gestures. Robot says the gesture name when recognized. The second experiment is done on the *test* dataset. Experiments consist of a number of trials with binary outcome; either the gesture is recognized or not recognized. A gesture is recognized if the dynamic hand gesture recognizer makes its first prediction correctly and within the gesture time frame. Experiments were performed by 10 different people from various nationalities

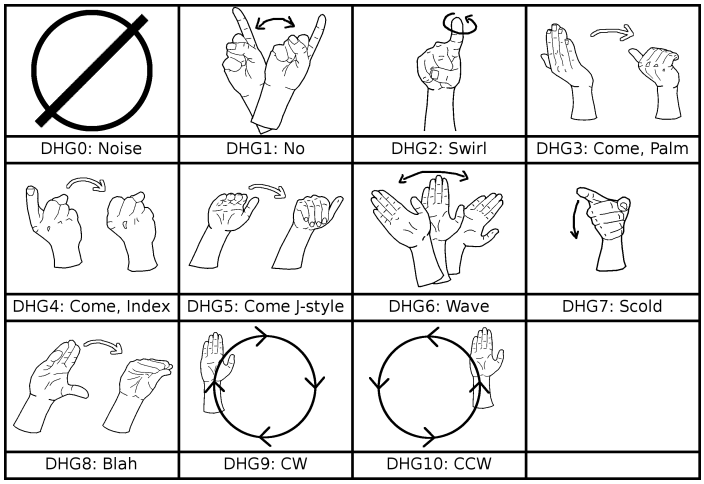

Fig. 2. List of dynamic hand gestures (DHG). A solid arrow represents movement, while the border arrow represents only a change in hand states

(Germany, India, Mexico, Pakistan, etc.) to account for possible variations that could affect the performance of the recognizer.

Table 1 shows mean accuracy and confidence interval of each gesture for both experiments. From the table, it can be seen that the mean accuracy of hand tracker is quite low which in response effects the dynamic hand gesture recognition system. If wrong tracking information from the tracker is ignored as shown in experiment 2 then the recognition rate goes to 100% which shows the potential of the system. If the hand tracker software greatly improves, we could expect the accuracy to increase to more than 93%, with a 90% confidence interval, as shown by the experiments. On average the system is able to recognize dynamic hand gestures with an accuracy of 86% along with some tracking errors.

4 Conclusion

This paper is about recognizing various dynamic hand gestures (DHGs) that can either be illustrators or emblems. The goal is to create efficient real-time DHG recognizers, that require little adjustment, and achieve high accuracy. The information flow process was described, from acquiring the depth image with the hand tracker position, to the results of classification. Dual classification strategy has been employed. In the first stage, frame level features are extracted and then classified using random forests. These classifiers are used to make sequence level feature vector which includes trajectory and velocity related features. The final classification is performed on this feature vector. Experimentation shows that the robot can recognize dynamic hand gestures and can achieve more than 90% accuracy if the hand tracker works efficiently.

Table 1. Accuracy statistics for two experiments: live experimentation with ROBIN and experimentation on recorded dataset (reduced tracking error). The results are expressed as the mean accuracy μ, and a 90% confidence interval $[p_L - p_H]$. As a reference, the mean hand tracker accuracy, and the expected value for gestures with less than 25% tracking error, are also shown. $\aleph$ denotes gestures

	Tracker		Experiment 1			Experiment 2		
DHG	μ	$E(\varepsilon < 0.25)$	μ	p_L	p_H	μ	p_L	p_H
$\aleph_1$	0.883	0.786	1.000	0.750	1.000	1.000	0.464	1.000
$\aleph_2$	0.891	0.929	0.750	0.444	0.936	1.000	0.316	1.000
$\aleph_3$	0.793	0.714	1.000	0.750	1.000	1.000	0.316	1.000
$\aleph_4$	0.771	0.706	0.750	0.444	0.936	1.000	0.316	1.000
$\aleph_5$	0.912	0.933	0.875	0.580	0.987	1.000	0.316	1.000
$\aleph_6$	0.443	0.154	0.875	0.580	0.987	1.000	0.316	1.000
$\aleph_7$	0.694	0.500	0.750	0.444	0.936	1.000	0.316	1.000
$\aleph_8$	0.845	0.750	0.625	0.327	0.863	1.000	0.316	1.000
$\aleph_9$	0.658	0.700	1.000	0.750	1.000	1.000	0.316	1.000
$\aleph_{10}$	0.780	0.556	1.000	0.750	1.000	1.000	0.316	1.000
$\aleph_{1-10}$	0.767	0.673	**0.863**	0.789	0.917	1.000	0.896	1.000

References

1. Celebi, S., Aydin, A.S., Temiz, T.T., Arici, T.: Gesture Recognition using Skeleton Data with Weighted Dynamic Time Warping. In VISAPP, pp. 620–625 (2013)
2. Ekman, P.: Movements with Precise Meanings. Journal of Communication, vol. 26, no. 3, pp. 14–26, (1976)
3. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Second Edition, ser. Springer Series in Statistics, Springer New York, (2009)
4. Knapp, M.L., Hall, J.A., Horgan, T.G.: Nonverbal Communication in Human Interaction. Cengage Learning, (2013)
5. Li, Z., Jarvis, R.: Real time hand gesture recognition using a range camera. In Australasian Conference on Robotics and Automation. 2009, pp. 21–27 (2009)
6. Ni, L., Aziz, M.A.A.: A robust deep belief network-based approach for recognizing dynamic hand gestures. In 13th International Bhurban Conference on Applied Sciences and Technology (IBCAST). Jan 2016, pp. 199–205 (2016)
7. Plouffe, G., Cretu, A.M., Payeur, P.: Natural human-computer interaction using static and dynamic hand gestures. In 2015 IEEE International Symposium on Haptic, Audio and Visual Environments and Games (HAVE), pp. 1–6 (2015)
8. ROBot-human-INteraction (ROBIN). http://agrosy.informatik.uni-kl.de/en/robots/robin/
9. Sohn, M.K., Lee, S.H., Kim, D.J., Kim, B., Kim, H.: 3D hand gesture recognition from one example. In 2013 IEEE International Conference on Consumer Electronics (ICCE). Jan 2013, pp. 171–172 (2013)
10. Tang, M.: Recognizing hand gestures with microsoft's kinect. Palo Alto: Department of Electrical Engineering of Stanford University:[sn], (2011)