Real Time Hand Gesture Recognition With 2 Kinect Sensors

Radu Paul Mihail
315 Davis Marksbury Building
329 Rose Street
University of Kentucky
Lexington, Kentucky
r.p.mihail@uky.edu

Nathan Jacobs
319 Davis Marksbury Building
329 Rose Street
University of Kentucky
Lexington, Kentucky
jacobs@cs.uky.edu

Judy Goldsmith
309 Davis Marksbury Building
329 Rose Street
University of Kentucky
Lexington, Kentucky
goldsmit@cs.uky.edu

Abstract

In this paper, we propose a robust static hand gesture recognition algorithm that makes use of two Kinect sensors. This will be used to control an avatar in a decision aid for rheumatoid arthritis patients who have a difficult time using a standard keyboard and mouse interface. The sensors are placed on the left and right sides of a target sensing area, easily set up in a doctor's office or waiting room. The Kinects provide a rich point cloud, out of which gestures from a known vocabulary are recognized in real time. We use 6 point cloud descriptors simultaneously and employ the majority rule voting scheme to pick a "winner" gesture in real time. We achieve rotation invariance by using part of the forearm as a good indicator of hand orientation and aligning the hand with the world coordinate system origin. We evaluate the performance of the recognition system under various motion and rotation conditions.

Contact author:
Radu Paul Mihail, r.p.mihail@uky.edu

Keywords:
gesture recognition, kinect, majority rule, voting, rheumatoid arthritis

Conference submitted to:
IPCV

1. Introduction

Rheumatoid arthritis (RA) is a potentially debilitating, life-long autoimmune disease that affects millions of adults around the world. Individuals suffering from RA face a daunting array of treatment choices, each with its own benefits and side effects. The challenge in making such choices is often difficulty in comprehending the consequences of a particular choice of medications and other treatments. To the best of our knowledge, decision tools for RA based on computer game technology have not been explored. The problem for RA patients is that the standard modes of interaction are often impossible because the disease limits range of motion and can cause physical deformities: simply manipulating a mouse or grabbing a Wii-mote may be impossible.

In this paper, we propose a real-time gesture recognition system that uses a pair of Kinect sensors to distinguish between static hand gestures. Our system is designed to be easily customized for individual users and to be rapidly configured in a doctor's office. Given the physical deformities and limited range of motion, each user may require a special set of gestures that are determined interactively by a therapist. Therefore, we minimize training time by using a lazy-learning algorithm to classify individual gestures.

In the context of our larger research program, this gesture recognition system will enable RA sufferers to control a character in a virtual environment. The goal is for them to use this system to visualize the outcome of a particular treatment plan by enabling them to perform actions, such as making a cup of coffee, which might be impossible for them in the physical world. For this task, our gesture recognition system might include gestures for grabbing, standing, walking, and placing-on-a-table to control avatar motions. Figure 1 shows examples from the library of natural static hand gestures we use in our experiments. The heart of our proposed system is a voting-based scheme that combines the results of 6 nearest-neighbor classifiers to determine the current gesture. The resulting system works in real time on a modest computer with average resources. An important feature of our method is that it does not require tracking or manual pose initialization; these techniques were deemed too brittle for our proposed users, and, as our results show, were not necessary. Instead, we classify individual gestures on a per-frame basis using a single point cloud, fused from two Kinect sensors, as input. The most computationally demanding part of our system is the component that manipulates the point cloud, namely a rigid-body transformation and a hand segmentation procedure. Once segmented, we convert the point cloud into a voxel-based representation and extract 6 different features.

Our results demonstrate that the proposed system can accurately classify a realistic set of gestures in real time. We demonstrate that the system can be rapidly retrained for new users. Future work will more extensively evaluate the system and integrate this gesture recognition system into a larger virtual environment to aid RA sufferers to make informed medical decisions.

1.1. Related Work

Recovering the full kinematic parameters of the skeleton of the hand over time, commonly known as the hand-pose estimation problem, is challenging for many reasons: high dimensionality of the state space, self occlusions, insufficient computational resources, uncontrolled environments, rapid hand motion and noise in the sensing device [3]. Erol et al. [3] provide a comprehensive review of research on this problem. Given our application, we focus on a special case of hand-pose estimation known as gesture recognition, in which discrete hand poses are detected. Solutions to this problem can be generally divided into appearance-based methods, depth-camera methods, and tracking-based methods. In the remainder of this section we give an overview of the gesture recognition literature.

Direct Appearance-Based Methods Athitsos et al. [1] propose a real-time gesture recognition system that uses a large database of synthetically generated images of hands in various configurations. The proposed approach relies on searching a database of tens of thousands of potential gestures. They use an indexing scheme, known as BoostMap, to decrease query processing time and thus enable real-time recognition. For their system, the input data consists of 2D images, out of which the hand silhouettes are extracted through skin color segmentation. Their method relies heavily on a relatively clean segmentation. Unlike typical appearance-based methods, where estimation is done from a limited number of viewpoints such as Ying et al. [8], Athitsos et al. [1] allows arbitrary views. [4] Hassan et al. propose a method using 2D images from a simple camera using the hand contour information and complex moments which are rotationally invariant. The complex moments and contour information are used as input to a feedforward neural network that classifies a vocabulary consisting of 6 gestures. They obtain accuracy of 86.38%, which is lower but comparable to our approach.

Direct Depth-Based Methods Gesture recognition has been revisited with the introduction of inexpensive depth cameras. Most similar to the current paper, is work by Suryanarayan et al. [7] in which they propose a gesture recognition system using a single ZCam camera from 3DV Systems. One limitation of their approach is their method of obtain invariance to hand rotation. Their method introduces a limitation on the types of poses that can be successfully distinguished. Our hand/forearm segmentation method, coupled with a similar PCA-based normalization

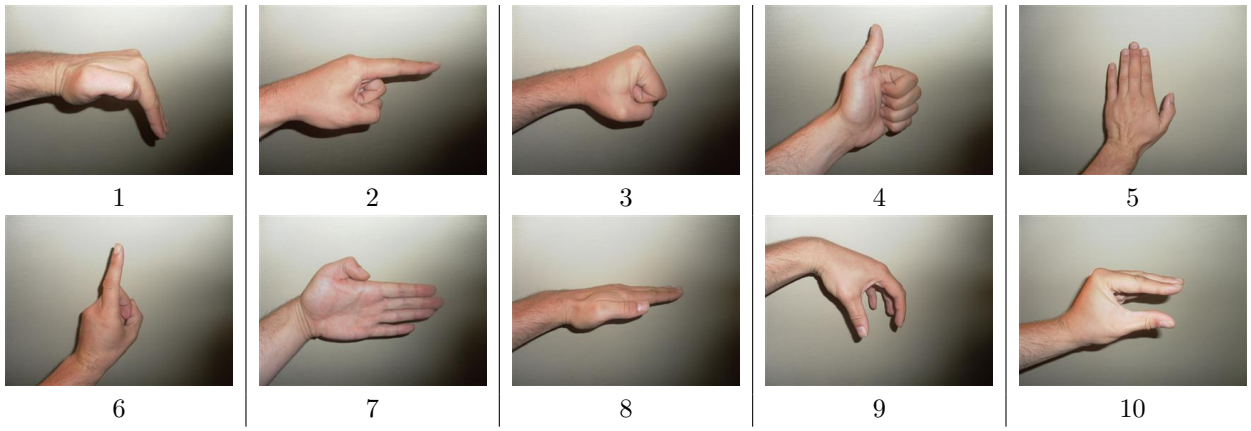

Figure 1: Gesture vocabulary.

scheme, enables our system to accurately distinguish between a more varied set of gestures. Zhou et al. [6] proposed a method to recognize hand gestures using a Kinect sensor. Their approach uses color, as well as depth information. The color is used to segment the hand from the rest of the environment, while the depth is used in a template matching algorithm where the dissimilarity measure they use is "Finger-Earth Mover's Distance" (FEMD). They claim an average of 90.6% accuracy on a dataset, but mention nothing about distortion or rotation invariance.

Tracking-Based Methods An alternative approach to gesture recognition involves first solving the hand-pose estimation problem. Once this is solved it is straightforward to determine the gesture based on geometric parameters of the hand. Hand-pose estimation methods often rely on tracking, where an initialization step has to be performed, and the pose at the current frame relies on knowledge about the previous frame. [5, 9] In such applications, if the system loses track, it can only be recovered using manual initialization. Unlike these systems, our approach estimates pose from a known vocabulary at every frame, independent of previous frames.

2. A Real-Time Gesture Recognition System

We propose a system that uses point clouds obtained from two consumer depth cameras to recognize gestures in real time. In this section, we first describe the intended use of the system and then give details of the physical and conceptual configuration of our system.

2.1. Usage Scenario

We designed the system to be robust and easy to set up in a doctor's office environment. The two Kinect sensors can easily be placed on a desk, and the system runs on an inexpensive machine. Our choices of gestures are familiar gestures that will be useful for our application. We designed this algorithm for natural interaction and control of an avatar through gestures. A simulation involves directing the avatar through hand gestures to perform certain activities.

Imagine that the user wishes her avatar to go to the kitchen, get a cup of coffee, come back, and put the coffee on a table. This may sound trivial to the reader, but for someone with advanced rheumatoid arthritis (RA), this may be impossible. Standing up and sitting down are painful, sometimes impossible. Holding the coffee cup in one hand requires a steady grip; we have heard patients describe walking across the room with a cup, and suddenly the cup and coffee are on the floor.

We want to show the user an avatar that attempts this sequence of actions. At one point in the process, the avatar will display the user's degree of RA and associated deformation. At another point, the avatar will display a possible effect of the medication (calculated according to the probabilities of effects in the medical literature). The user, however, will not suddenly get better in the doctor's waiting room. So control of the avatar must be doable, even with RA-inhibited or deformed hands.

The user will be able to gesture with a finger pointed up to stand, a horizontal palm to turn, etc. These gestures will all be user-tested for comfort and performability. We are preparing an IRB application for a further study.

2.2. Data Acquisition

We create a point cloud using two Kinect sensors. In this work, we ignore color information, and down-sample by reading every third pixel in the depth map to increase the per frame processing speed. There are two main reasons why we can ignore color: first, the hand is relatively flat textured and second, we can't assume consistent light-

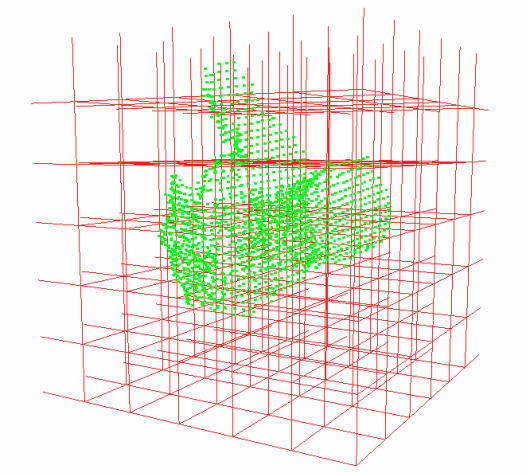

Figure 2: An example point cloud of gesture captured by our system when inevitable occlusion is overcome by using two Kinect sensors.

ing conditions. In fact, our system works in complete darkness. In our experiments, the yaw angle for which the device has a motor is set at the default value, which results in an orientation parallel to the table top on which the sensors are placed. An angle of 45° provides the most information when performing gestures that have inevitable occlusion as depicted in Figure 2, because within a certain distance from the cameras, each of the two sensors "sees" half of the hand, hence capturing more structural information.

We map raw depth values for each Kinect, which are 12 bit integers, to metric 3-space using Nicholas Burrus' formula [2]. Because the two video streams come from two sensors placed in different places, we have to perform a calibration step (see Appendix **??**), which rotates and translates the coordinate system of one camera to match that of the other camera. We do not use color information, but we acknowledge that it could provide additional useful information for solving the hand pose estimation problem.

2.3. World Coordinate System

The rotation around the Y-axis previously determined in the calibration step (Θ) is part of the transformation that maps one Kinect's coordinate system to the other Kinect. We create a world coordinate system, where the depth axis points exactly in between the two Kinect sensors. This new depth axis coincides with the bisector line of the normal vectors with the origins at the center of the camera sensors as depicted in Figure 3 by Zworld. Angle Θ is known from the calibration step, so in practice we rotate the point clouds around the world coordinate system Y-axis by $\frac{\Theta}{2}$ and $-\frac{\Theta}{2}$ (clockwise and counterclockwise respectively). We create this coordinate system because users will interact with the system by placing their hands between the two sensors with

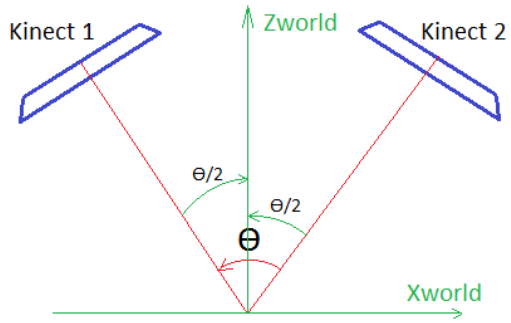

Figure 3: World Coordinate System depth axis.

the arm pointing along our world coordinate system depth Z-axis. This step is useful in creating a descriptor (Section 3.3) where the real depth coincides with the line from wrist to the tip of the fingers. Having an intuitive world coordinate system is invaluable if we need to extract the spatial position and orientation of the user's hand when performing a certain gesture. In our proposed application, a user may navigate the virtual 3D world by pointing with their index finger to determine what direction the avatar will walk next. When recognized, this gesture will cause the avatar to turn her head in the direction where the user's finger is pointing. After a direction has been established, another gesture can be used to make the avatar walk forward or back up. If, for example, the user wishes to pick up an object in the virtual world, the relative position of the hand in our world coordinate system can be translated into that of the virtual world so the avatar can move her hand in a 3D position indicated by the user. The relative position of the hand is the center of mass of the point cloud derived from the segmentation process.

3. Gesture Recognition

We formulate the gesture recognition problem in the context of this paper as follows: given a snapshot of a hand configuration (a point cloud), decide which gesture from a repertoire of known gestures has been performed. Using a cluttered point cloud requires a segmentation process, to isolate the hand from the rest of the scene. We use a simple segmentation process described in the next section. The simplicity of the segmentation is acceptable due to the way we engineered the system to allow for the hand to be the closest object to the Kinect sensors.

3.1. Segmentation

Given the final point cloud Γ=$\{p_1, p_2, ...p_n\}$ in the world coordinate system mentioned above, we need to extract the points that belong to the user's hand. We assume that, during usage, the hand will be the closest object to the two Kinect sensors. We extract closest point $p_{closest}$ =

(X, Y, Z). We then search Γ for a subset of points $\Gamma' = (hp_1, hp_2, ..., hp_n)$, where the following conditions hold.

1. $p_{closest}.Z < hp_n.Z < p_{closest}.Z + 0.30m$
2. $p_{closest}.X - 0.20m < hp_n.X < p_{closest}.X + 0.20m$
3. $p_{closest}.Y - 0.20m < hp_n.Y < p_{closest}.Y + 0.20m$

The subset Γ' is guaranteed to be contained in a box with volume $0.30m * 0.20m * 0.20m = 0.048m^3$. We chose 0.30m for the depth because it captures part of the forearm, which we use to achieve rotation invariance, process described in section 3.2. The rotations with which we wish to achieve invariance to are about the X-axis and Y-axis of the world coordinate system. The values 0.20m for width and height of the bounding box, were determined empirically to ensure it can contain hands of various sizes. The second reason was the need to ensure the box can contain a hand and part of the forearm rotated along either X or Y-axis, which causes an increase in volume of the bounding box. Assuming the segmentation process completes successfully, Γ' contains points pertaining to the hand and part of the forearm.

3.2. Rotation Invariance

To achieve rotation invariance, we assume the forearm to be a strong indicator of the hand orientation. We find the principal component of this set and align Γ' along the X and Y components of the world coordinate system. It is true that gestures such as "Palm Front" and "Attention" will affect the principal component vector, and thus final rotation, but the bias is consistent across users, so it can be ignored. We call the angle that the principal component is rotated along the Y-axis of the world coordinate system α and the rotation angle along the X axis is β. This results in invariance to the two rotations. Unconstrained rotation along the Z axis is important because the semantics of the gestures depends on it, e.g., gestures 7 and 8.

3.3. Descriptor

To discriminate between gestures that users perform, we require a fast classification algorithm to process every frame of the combined streams. Therefore, designing a system that is computationally feasible on an average machine is important due to a requirement of cost efficiency to implement and deploy systems in various settings. In Section 3.1 we describe the segmentation process used to isolate hand and forearm points from the rest of the background and align it with the world coordinate system to achieve rotation invariance, and we named this set Γ'. The forearm points were used for alignment purposes, and can now be discarded. We eliminate them by "trimming" Γ' of the trailing 15cm volume.

In order to build a gesture vocabulary and implement a classification algorithm, we need to compute a descriptor for Γ'. We use six point distribution histograms for Γ'. We subdivide the volume into $6^3, 8^3$ and 10^3 voxel spaces of equal size, distributed evenly.

For each such division, we produce two descriptors; one where each dimension of the descriptor corresponds to the point count in each voxel space and the other is a binary value describing whether there is one or more pixels in each voxel space. We use Ψ to denote a count histogram and ψ to denote a binary histogram. The final descriptors are $\Psi_{6^3}, \psi_{6^3}, \Psi_{8^3}, \psi_{8^3}$ and Ψ_{10^3}, ψ_{10^3}. Since the hand is extracted from the world coordinate system, the position is important to our application. We determine the absolute position of the hand by computing the center of mass for the point cloud Γ' and name it Ω – a vector in 3-space.

3.4. Recognition

We use a nearest neighbor classifier in combination with majority rule voting scheme to recognize unknown gestures. For each unknown gesture, we compute the six descriptors and map the points to the feature spaces of each descriptor. We compute distances between all cluster centers. The recognition process for an unknown gesture involves computing the squared distance in each feature space between it and known gestures in the vocabulary,

$$d^2(\Psi_{unknown}, \Psi_{known}) = \sum_{n=1}^{i} (\Psi_{unkown}[i] - \Psi_{known}[i])^2$$

where i is the dimensionality ($6^3, 8^3$ or 10^3).

We retrieve the top two matches, and the threshold that determines whether the gesture is known or not is twice the distance between the top two cluster centers. For each feature space, the nearest neighbor is retrieved. The results may not be the same for each descriptor due to noise, pose and the user making slight variations in the way the gesture is performed. We view this as a voting problem with 6 voters where each vote is a recognized gesture.We noticed that the majority of descriptors agree on the correct gesture, so we use the majority rule voting scheme to pick the "winner" candidate gesture. In the case of a tie or if all are different, we classify the gesture as unknown.

4. Evaluation

To evaluate the effectiveness of our approach, we designed the following experiment: first we build a training set by asking users (co-authors) to perform the 10 gestures in an arbitrary order and repeat the process 3 times. We collected a total of 30 data points per user in each of the feature spaces. In the evaluation sessions, training data was limited to the user testing the system. We did not mix training data among users because of the wide variation in the way different people perform a single gesture. Furthermore, the

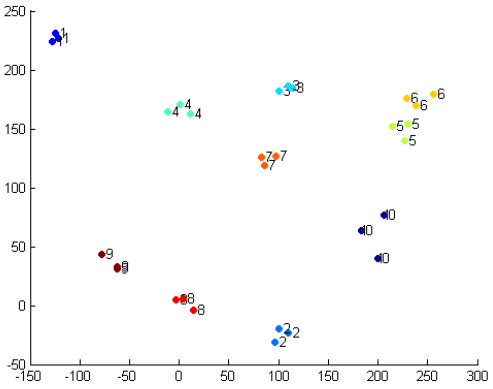

Figure 4: Each of the above points is a gesture in the set Ψ_{6^3}. We used PCA to reduce dimensionality and show the well separated clusters in the feature space.

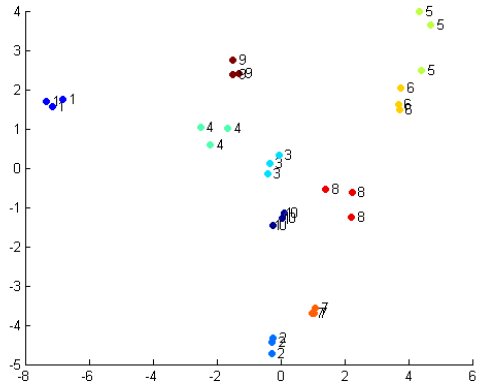

Figure 5: Set of 3 ψ_{10^3} per gesture.

system is intended to work with hand deformities and restricted joint movements.

After training data was collected, users were asked to perform gestures as prompted by the application for 6 seconds with 3 second breaks between gestures. During the break, the application prompts the next gesture that is to be evaluated. We recorded the following information for every frame: ground truth gesture, majority rule recognized gesture, recognized gesture for each descriptor, time in milliseconds to process raw kinect data (transformations, segmentation, rotation based on principal component) and finally angles α and β found as described in Section 3.2. We collected angles to determine whether there is a correlation between hand orientation and recognition performance.

4.1. Training Data

The training process consists of computing a set of descriptors Ψ and ψ for each gesture, for a total of 6 data points. We asked users [1] to participate in the training process by performing the 10 gestures in a specific order. They repeat the 10 gestures 3 times with pauses in between. The effect of an arbitrary ordering is more variation compared to repeating the same gesture several times before moving on the next. To demonstrate this variation, we reduced dimensionality using PCA for each descriptor to visualize the clusters they form, see Figures 4 and 5. Notice how similar gestures form clusters that are closer.

As an example, gestures 5 and 6 ("Palm Front" and "Attention") are similar, as seen in the graphs. Gestures 2 and 5 ("Point Straight" and "Palm Front") are significantly different, placing them farther apart in the feature space.

To further illustrate the effectiveness of the descriptor, we compute distances between two random sets of gesture examples and graph the results in Figure 6. As we expected, the main diagonal has low values, which means that our descriptor is discriminative in ideal conditions. In practice, we have to take into consideration the limitations of the Kinect sensors. As an example, the minimum depth sensing distance is 3 feet. If a user gets closer than the minimum distance, the end result is points being dropped from the cloud, which negatively affect the recognition system. Furthermore, because the depth sensor is based on a pinhole camera with inexpensive optics, there is more distortion near the edges of the depth map. In our setup, unless users are restricted to a relatively small space, the detected hand may be near the edge of one or both of the depth maps, which also negatively affect the recognition performance. In the experiments below, we evaluate the performance of our suggested approach by performing 3 experiments. In the first experiment the user does not move the hand, in the second one the user moves her hand arbitrarily and in the third experiment, the user moves her hand in a plus like pattern.

4.2. Experiments

Experiment with user not moving her hand In this experiment, the user performed a set of 10 gestures without moving her hand. Figure 7 shows the comparison between the descriptors. The thick red line represents the majority rule gesture accuracy. There were no incorrect classifications in this experiment over a total of 1477 frames collected. We recorded the average α and β for this experiment and graphed them in Figures 8 and 9. Higher normalization angles resulted from the bias that some gestures cause; e.g., gestures 5 and 6.

We also recorded the average processing time for each frame. The processing times are inclusive of raw data acquisition and transformation, segmentation and recognition.

[1] The users for this study were the authors. We are preparing an IRB application for a wider study.

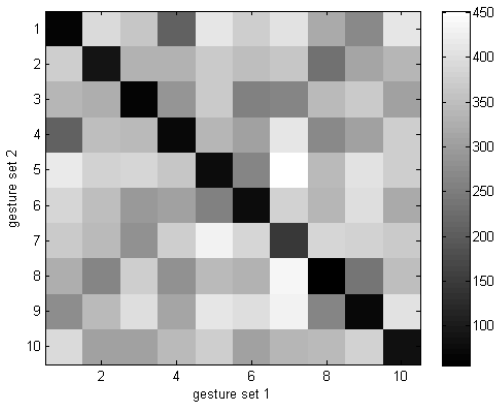

Figure 6: Average distances between pairs of gestures using Ψ_{6^3} descriptors.

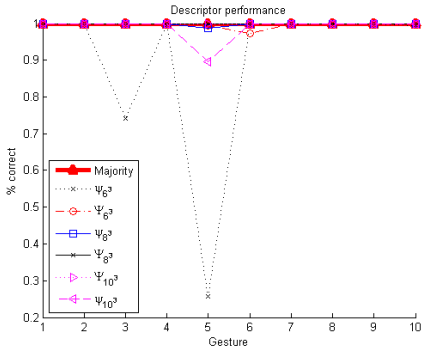

Figure 7: In the above figure, the performance of the six descriptors is graphed. Most descriptors performed well, while the majority vote achieved 100% recognition rate.

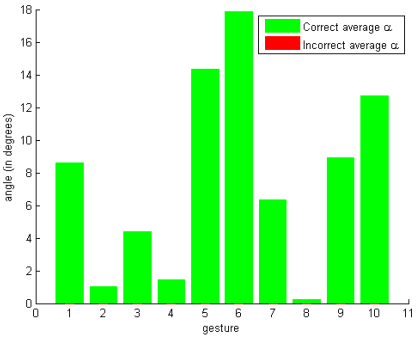

Figure 8: In the above figure, the average orientation angle α is graphed for both correct and incorrect classifications by majority rule.

The average processing time per frame was found to be around 35ms on a dual core Pentium D processor running at 2.8 Ghz, without any processor specific optimizations. We

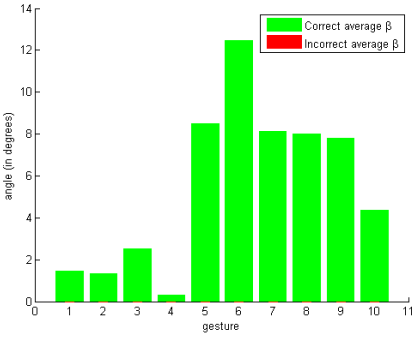

Figure 9: Average orientation angle β.

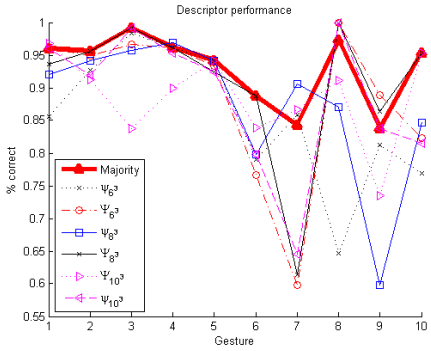

Figure 10: In the above figure, the performance of the six descriptors is graphed when the user moves their hand in a circular pattern. The recognition rate is lower due to orientation normalization.

wrote the prototype in C#.

Experiment with user making a circle pattern while gesturing This experiment is similar to the first, except the user was asked to move her hand arbitrarily. The total number of frames processed for all gestures was 1274, out of which 1187 were classified correctly (93.17%). Figure 10 shows the recognition accuracy of individual classifiers and the majority rule vote. There is a weak correlation between higher normalization angles and recognition accuracy as seen in Figures 11 and 12. We present the conditional probability distribution of misclassification given angle α in Figure 13.

Experiment with a plus like motion In this experiment, the user was asked to perform the set of gestures as in the first two experiments, but he was asked to move his hand in a plus like pattern. We show a confusion matrix in Table 1. In figure 14, the performance of individual descriptors and the majority rule is graphed. A total of 1654 frames have been processed, out of which 1602 have been correctly

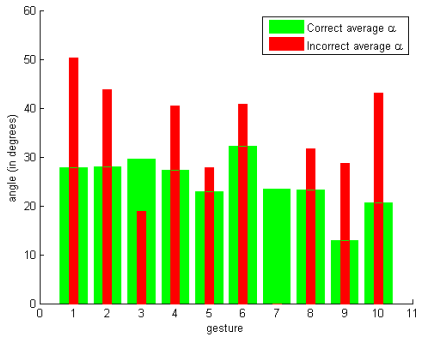

Figure 11: The detected average orientation angle α is graphed for correct and incorrect classifications. Note that there is little correlation between normalization angles and recognition accuracy.

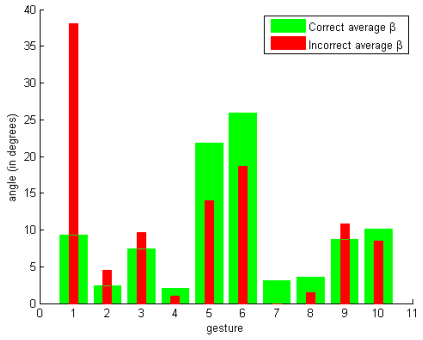

Figure 12: Detected average orientation angle β.

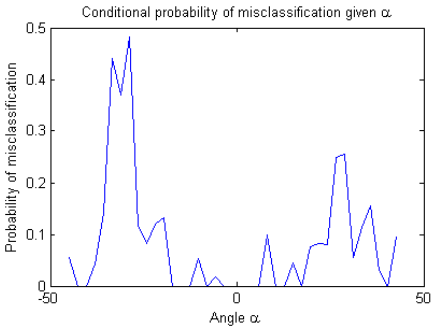

Figure 13: In this figure we represent the conditional probability distribution of misclassification given angle α. We noticed that rotation about the Y-axis of the world coordinate system produces the most classification errors.

classified (96.86%).

Comparison to other methods We evaluated the effectiveness of our system under various rotation and motion conditions. Similar work has been done by Suryanarayan et al. [7], however, there are differences that prevent a comprehensive comparison. First, they use a single time of flight (TOF) camera to augment the 2D images in their gesture vocabulary with depth data. They evaluate their system using a vocabulary of 6 gestures, however, it is unclear the motion and rotation constraints they imposed on their evaluation data. They average slightly under 90% accuracy across all 6 gestures. Our system accuracy is 100% when no motion/rotation is performed, however, we analyzed extreme cases of rotation which were found to decrease performance. Our system outperforms the one proposed by Suryanarayan et al. [7] under mild rotation conditions. Motion will decrease performance when the user moves his/her hand fast enough to pass the limitation of the devices' frame capture rate of 30 per second.

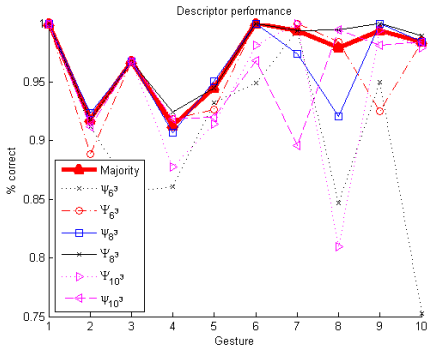

Figure 14: Descriptor performance for the third experiment.

5. Conclusion

In this paper we presented a novel algorithm for the recognition of static hand poses using two Kinect sensors. We describe how to merge two depth streams from sensors placed at an angle which captures 3D points belonging to a user's hand. We used a trivial segmentation method for the extraction of hand points and a used PCA to achieve rotation invariance with respect to the X and Y axis of the world coordinate system. The algorithm relies heavily on a clean segmentation. We use multiple descriptors computed from the point cloud and apply majority rule voting scheme to improve the recognition process. The algorithm is evaluated and we present detailed results that can be used for future work in gesture recognition using Kinect sensors. We found that hand orientation has a slight negative impact on the recognition performance of our algorithm, hence a shortcoming, however, this is mainly due to the segmentation process and distortion near the edges of the depth maps. In future work, we will improve the segmentation process to allow for a less constrained interaction environment.

	Predicted gesture label										
	1	2	3	4	5	6	7	8	9	10	Unknown
1	0.9181	0	0	0	0	0	0.0058	0.0058	0	0.0234	0.0468
2	0	0.9673	0	0	0	0	0	0	0	0	0.0327
3	0	0.0058	0.9128	0.0581	0.0058	0	0	0	0	0	0.0174
4	0	0	0	0.9448	0	0	0	0.0123	0	0.0245	0.0184
5	0	0	0	0	1.0000	0	0	0	0	0	0
6	0	0.0065	0	0	0	0.9935	0	0	0	0	0
7	0	0	0.0159	0	0	0	0.9788	0	0	0	0.0053
8	0	0	0.0063	0	0	0	0	0.9938	0	0	0
9	0	0	0	0	0	0.0053	0	0	0.9842	0	0.0105
10	0	0	0	0	0	0	0	0	0	1.0000	0

Table 1: Confusion table. Each row represents a gesture that was evaluated.

6. Future Work

The results we obtained in our work can be improved by employing a better segmentation process. Our segmentation method is simple and effective when the user's hand is the closest object to the Kinect sensors. However, this requires open space near the sensors which may be difficult to obtain in a waiting room or doctor's office. We will improve the segmentation and test our system in a clinical setting with patients. This work can also be used as a preprocessing step to solve the hand pose estimation problem by discretizing possible hand joint configuration spaces, thus limiting the search problem.

References

[1] V. Athitsos, H. Wang, and A. Stefan. A database-based framework for gesture recognition. *Personal and Ubiquitous Computing*, 14(6):511–526, 2010. 2

[2] Nicholas Burrus homepage - kinect calibration. webpage, November 2010. URL: http://nicolas.burrus.name/index.php/Research/KinectCalibration. 4

[3] H. A. Erol, G. Bebis, M. Nicolescu, R. D. Boyle, and X. Twombly. Vision-based hand pose estimation: A review. *Computer Vision and Image Understanding*, 108(1-2):52–73, 2007. 2

[4] H. Hasan and S. Abdul-Kareem. Static hand gesture recognition using neural networks. *Artificial Intelligence Review*, pages 1–35, 2012. 2

[5] S. Lu, D. N. Metaxas, D. Samaras, and J. Oliensis. Using multiple cues for hand tracking and model refinement. In *CVPR (2)*, pages 443–450, 2003. 3

[6] Z. Ren, J. Meng, J. Yuan, and Z. Zhang. Robust hand gesture recognition with kinect sensor. In *Proceedings of the 19th ACM international conference on Multimedia*, pages 759–760. ACM, 2011. 3

[7] P. Suryanarayan, A. Subramanian, and D. Mandalapu. Dynamic hand pose recognition using depth data. In *ICPR*, pages 3105–3108, 2010. 2, 8

[8] Y. Wu, T. S. Huang, and T. S. Huang. View-independent recognition of hand postures. In *In CVPR*, pages 88–94, 2000. 2

[9] Y. Wu, J. Y. Lin, and T. S. Huang. Capturing natural hand articulation. In *ICCV*, pages 426–432, 2001. 3