Object Transfer Between Humans and Robots

Benjamin Jaeger, Benjamin Phillips, Cornell University, May 2011

***Abstract*—The goal of our project is to transfer objects between a user's hand and a robotic arm. The robot will recognize when a user is attempting to pass it an object and then will try to grab the object out of the user's hand. The project consists of three parts: recognizing the user's handing gesture, locating the object, and grasping the object. To recognize the user's handing gesture we take skeleton tracking data and employ multinomial logistic regression along with a Hidden Markov Model.**

I. Introduction

Historically, real-time human-robot interaction in 3D space has been difficult. A single camera can only accurately provide 2 dimensions and generating the 3rd dimension requires extra intensive analysis. An alternative is to use a camera with depth perception. Until recently, there haven't been any cheap and easy to use 3D vision rigs. The Microsoft Kinect Sensor is a relatively new device that provides advanced human skeleton tracking capabilities through the Openni library at a very low cost. For our project we attempted to leverage the power of the Kinect Sensor to see if it is appropriate for real-time recognition of human handing gestures and for real-time interaction with people via a robotic arm.

A project completed by Aaron Edsinger, MIT had a similar goal as ours: to transfer objects between a human hand and a robotic arm. However, he used a single camera mounted on a pan/tilt rig and used a custom algorithm to generate depth perception [1]. We hope that the Kinect will lend to a much easier implementation. Yanghee Nam et. al. used a Hidden Markov Model to recognize gestures in 3D space [2]. However, they simplified 3D gestures into 2D with plane fitting before applying the HMM. The Kinect allows us to consider all three dimensions.

Our project consisted of three parts: recognizing the user's handing gesture, locating the object, and grasping the object, all in real time. First, we recorded a data set of positive and negative handing examples. The data set consists of joint positions provided by the skeleton tracking library. To model a gesture, we broke it down into finite states. We trained a multinomial logistic regression classifier with joint angles and angular velocities as features and used it to provide real time state estimates. Then to enforce order on the output of the logistic classifier we trained a Hidden Markov Model, and employed Viterbi's algorithm to estimate the current state. Once a gesture is detected, the next step is to locate the hand. The Kinect and the skeleton library make this task fairly simple. We simply transform the location of the hand joint from the reference frame of the Kinect into the reference frame of the robot base. Unfortunately, we were unable to test our algorithm on the arm due to time constraints. However, if we had time, we would have sent the arm to the given coordinates and closed the arm once an object was positioned between it's grippers.

Fig 1. Microsoft Kinect Sensor

II. Hardward and Logistics

A Microsoft Kinect Sensor along with the Openni Skeleton Tracking Library was used to generate a training data set for the classifier and for real-time gesture classification. When creating the data set the Kinect sensor was attached to a desktop computer running ROS Diamondback Unstable on Ubuntu 10.10. We did not ultimately get to run our classifier on a robotic arm. The arm that we planned to use was a Barrett Arm with a Kinect sensor mounted above the base.

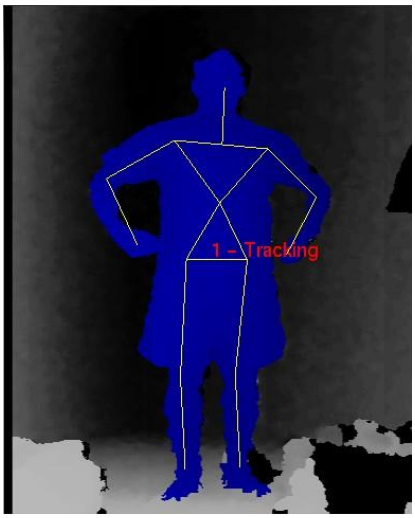

Fig 2. Openni Skeleton Tracking Library tracking a person using 3D depth input from the Microsoft Kinect Sensor

III. Data

We recorded 3D coordinates of the user's joint positions at each frame given by the Openni skeleton tracking library. The rate of image capture on the Kinect is 25 fps. The

library calculates (X,Y,Z) joint coordinates with the origin located at the camera.

Our data set has 8112 frames corresponding to roughly 5.4 minutes of video. It includes 76 sample handing gestures from two different individuals. We divided the handing gesture into 4 discrete states: base (ie: no gesture), beginning, middle, and end. While recording the data we marked in real time whether the user was in the process of making a handing gesture or not. A low signal means no gesture and a high signal means gesture. After collecting the data, we went through it and evenly divided the positive example frames into the start, middle, and end states. Due to the speed of a gesture, marking the frames with individual states in real time was not feasible.

IV. Training

We used MATLAB's multinomial logistic regression mnrfit() function to train our program to recognize a handing gesture. Logistic regression training takes as input a set of features X and their respective states S. The training outputs a set of coefficients B_s for each state to use with a logistic classifier. The classifier's output is governed by the equation

$$P(y_i = s) = \frac{\exp(X_i \cdot B_s)}{1 + \sum_j X_i \cdot B_j}$$

This output should be interpreted as a probability for each state, where for the state space S, $\sum_s P(y = s) = 1$. For our case we decided to discretize this output by using a winner-takes-all approach. Thus the ultimate output from our logisitic classifier is the state with the highest probability.

For the logistic regression features, we chose to use angles between various body parts and also the rate of change of those angles (angular velocity) to capture the time-based properties of a gesture. The angle between three body parts A, B, and C is taken to be the angle between BA and BC. Angles and their respective velocities are very simple to compute given the skeleton data so they are appropriate for real time classification. After some heuristic experimentation, the features we settled on were the following angles (1) right-shoulder, right-elbow, right-hand (2) right-hand, right-hip, right-shoulder (3) left-hip, right-elbow, left-shoulder (4) right-elbow, right-shoulder, torso.

For statistical purposes we split our data set into a training set and test set for cross validation. The training set contains 70% of the data while the test set contains 30% of the data. See the "Accuracies" section for how our algorithms performed on the test set.

V. Real Time Recognition

Our real time analysis algorithm mirrors the training process. For each frame arriving from the Kinect sensor, the features are calculated from the skeleton data. The logistic regression classifier, now with known parameters, determines the state with the highest probability.

An initial shortcoming with the the real-time analysis is that the classifier was sometimes determining that the user was at the end state without going through the beginning and middle states. For example the sequence of states {..., base, end, base, ...} was allowed while the desired sequence was {..., base, start, middle, end, base, ...}. This shortcoming was not apparent in our training set because all of our examples contain the full sequence {..., base, start, middle, end, base, ...}.

To solve this problem, we employed a Hidden Markov Model to enforce a linear progression of states. To train the HMM, we used MATLAB's hmmestimate() function with the observations Y being the output from the regression - the state with the highest probability, and the true state X as the marked state in the original training data. Finally, we employed the Viterbi algorithm to determine the most likely current state given the trained emission and transmission matrices calculated from the HMM.

A distinct feature of a handing gesture is that the user makes the gesture and then leaves their arm in an extended pronated position while waiting for the object to be taken. If the user moves their hand in an unusual motion and finishes in the waiting position, then eventually the Viterbi algorithm believes that the user is most likely trying to hand over an object even though the logistic classifier never output the start and middle states. This happens if the user keeps their hand in the waiting position for a long enough period of time that the probability of being in the end states overwhelms the Viterbi's consideration of the going to the proper state sequence {beginning, middle, end}. For the states other than the end/waiting state, the joint angular velocity features enforce time dependence on our model. As an example, for the middle state to be continuously recognized, the user would have to be continuously moving his arm forward (which is only possible if he moves his body also). However, since the velocity of the waiting position is zero, the velocity is irrelevant. To solve this problem, we manually tweaked the trained emission matrix values for when the end state is observed. Let X be random variable representing the true state and Y be the observation. We lowered Prob(X = end | Y = end) and increased Prob(X = base | Y = end). This means that if the user's hand is in the end state but did not make the gesture, then the Viterbi algorithm can recognize that there is a good chance that no gesture is happening.

Finally, we increased the transmission matrix value for going from the base state to the start state. This value represents the quantity "How often will the user make a handing gesture in a given amount of time?" This value needs to be independently tweaked as it cannot be accurately determined from the training data. In the training data, the user makes handing gestures at relatively periodic intervals whereas in the real world handing gestures are at varying time intervals and frequencies. Increasing the value increases the sensitivity of the model to the potential start of a handing gesture.

Once the Viterbi algorithm arrives at the end state, ie. the user has completed the handing gesture, the position of the user's hand is recorded. Using a transformation matrix based on the geometric setup of the Barrett arm with respect to the Kinect sensor, we transformed the coordinate of the user's

hand from the reference frame of the Kinect into the reference frame of the robot base. We planned on using the Barrett's arm inverse kinematics library to move the arm to this position.

We took various distance readings from the Kinect skeleton tracker ($\sqrt{x^2 + y^2 + z^2}$) to verify that its accuracy is appropriate for human-robot interactions in 3D space. Our measurements ranged from around 820 - 1420 mm. This range is appropriate for interacting with the Barrett Arm. At 820mm and less, the Kinect starts to get double vision (similar to when a we try to focus on an image to close to our *pair* of eyes). However, farther away, the percent error stays below 5%. A 5% error at 1420mm is only an error of 71mm which is reasonable.

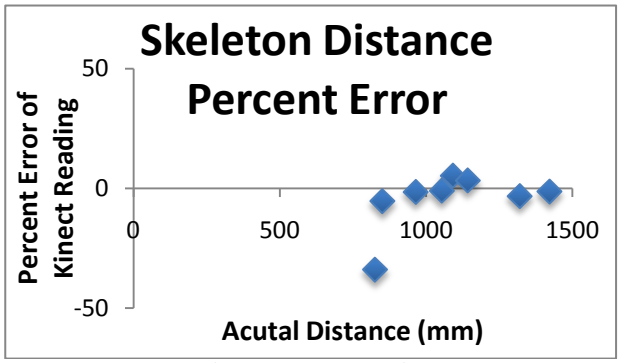

Fig 3. Percent error of the distance reading from the Kinect Sensor

VI. Accuracies

We used a cross validation data set, different from the training set, to measure accuracies for both the logistic classifier and the HMM. We measured both the type I error, the rate of false positives, and the type II error, the rate of false negatives. For convenience, we also report the accuracy rate which is really just 1 - false negative rate.

For logistic regression, we report measurements for each state individually since each state occurs at different frequencies.

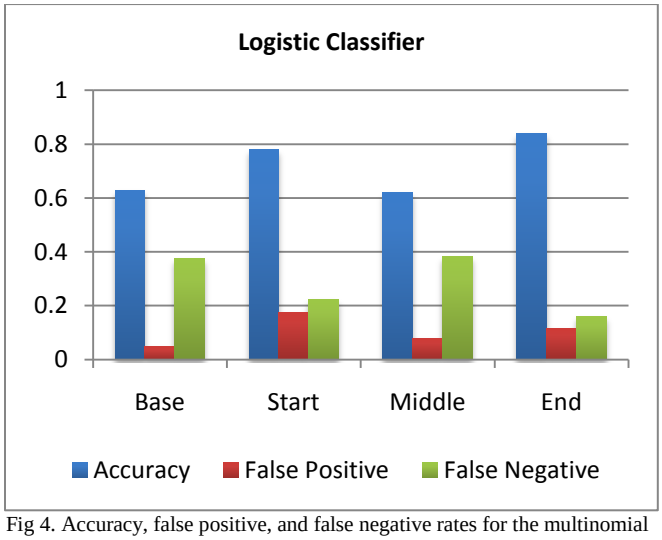

Fig 4. Accuracy, false positive, and false negative rates for the multinomial logistic classifier

We also used the output of Viterbi to measure statistics for the recognition gesture as a whole. We defined a true positive as a gesture occurring (ie. Viterbi outputting the end state) within a frame window of the marked end of the gesture. We used a window of 8 frames. A false negative then corresponds to no gesture occurring during the window. We then calculated the number of false positives = true total number of gestures - number of true positives. For percent false positives, there is no clear value to use as the denominator. Thus, we used (the number of frames in the data set marked that no gesture is occurring / the window length) as the denominator. Using this metric our HMM achieved 98.1% accuracy with a false positive rate of 4.78% and a false negative rate of 1.89%. While these numbers are very high, it's important to keep in mind that the training set may not accurately represent all real world scenarios. In practice, the HMM performs very well in real time though it is hard to get an actual statistic on it.

VII. Experiments and Limitations

1) Initially, we classified the frames in our data set as either base or handing (user is in the process of making a handing gesture). We trained the log classifier to recognize the start of a handing gesture. Unfortunately, the classifier was of limited utility because there was no way to differentiate between the start of a gesture and when the user is actually ready for an object to be grabbed. It was also very noisy. For these reasons we decided to generate a new data set with more finite states.

2) When we first tried to train a logistic regression classifier using four states, it got very poor results. The accuracy rates for the beginning and middles states were 0% while the accuracy rate for the end state was 12.9%. The accuracy rate for the base state was 98.3%. Upon closer inspection of the individual state probabilities from the classifier, it was clear that while the probabilities for the beginning, middle, and end states would change appropriately when given a sample of the proper state, the probability for the base state was dominating the output and thus was almost always being chosen by our winner-takes-all heuristic. Rather than chose a more complicated heuristic, we decided to try adjusting the training set to have an equal number of each state. This worked out well (see "Accuracy" section).

3) Currently the classifier only recognizes the handing gesture for the right hand.

4) We tried applying a mean filter over time to the joint position data, in an attempt to remove sensor noise. This actually decreased the accuracy of the algorithm. The most likely explanation is that the Openni library already removes noise.

5) We initially only used angles as input features to the logistic classifier. We experimented with including angular velocity features as well and observed much better results. For comparison, the results without angular velocity are presented below.

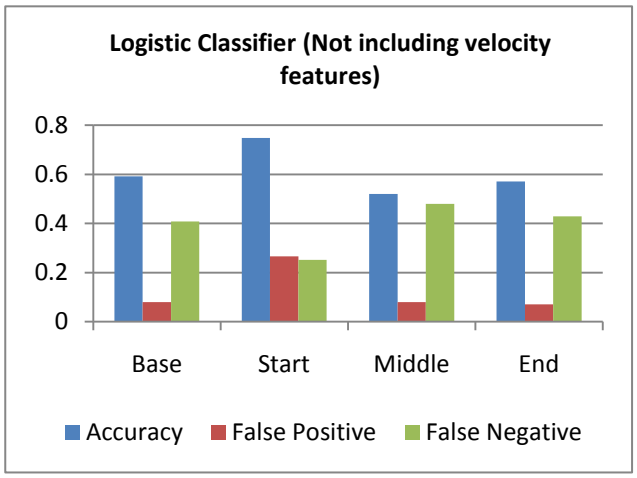

Fig 5. Accuracy, false positive, and false negative rates for the multinomial logistic classifier trained without angular velocity features

VIII. Conclusion and Future Work

The Microsoft Kinect Sensor along with the Openni Skeleton Tracker provide a new way of getting accurate and reliable 3D input to track people's movements. Combined, they greatly simplify the classic and difficult problem of gesture recognition. By utilizing both along with a Hidden Markov Model implemented on top of a multinomial logistic classifier we were able to develop a robust algorithm that could accurately and reliably recognize human handing gestures in both time and space. The algorithm is generic and can be trained to recognize any sort of body gesture visible from the skeleton tracker. While the feature set that we chose for the handing gesture was picked and evaluated by hand, when considering many different gestures, a more robust method would have been to use an optimization method such as simulated annealing for determining which set of features produce the best logistic classifier. Furthermore, our algorithm can easily be expanded to recognize multiple types of gestures at the same time. The state space would have to be increased so that each gesture has its own beginning, middle, and end state. Finally, if we had more time we would have liked to implement our algorithm on the Barrett Arm to demonstrate robot-human interaction.

IX. Acknowledgements

We would like to thank Colin Ponce for his advice throughout our work on this project. We would also like to thank Professor Ashutosh Saxena for advising us on our project and for teaching us machine learning and robotics.

References

[1] A. Edsinger and C. C. Kemp. Human-Robot Interaction for Cooperative Manipulation: Handing Objects to One Another. In Proceedings of the 16th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN), 2007.
<http://www.hsi.gatech.edu/cckemp/edsinger_kemp_roman2007.pdf>

[2] Nam, Y., Wohn, K.: Recognition of Space-Time Hand-Gestures using Hidden Markov Model,
ACM Symposium on Virtual Reality Software and Technology, Hong Kong, pp.51-58.(1996)
<http://citeseer.ist.psu.edu/viewdoc/download;jsessionid=DEE4DEDB30710A327156CF9F00722FDD?doi=10.1.1.42.4147&rep=rep1&type=pdf>