Annoyance Detection Using the Microsoft Kinect

Sam Sinensky, David Diner – Cornell University

***Abstract*—As robots become a normal part of life, ensuring a harmonious interaction between humans and robots will become an important consideration when designing and deploying robots. In our project, we attempt to design a robot than can sense if it is annoying a person, so that it can plan a path through a room that causes minimal annoyance to humans.**

A significant challenge for this project was collecting and properly labeling training data. We could not collect poses or gestures because of the inherent ambiguity associated with the poses. Instead we chose to record a short video clip of a person. We do annoyance inference based on changes in position, velocity, and acceleration. We used an SVM classifier on 250 datasets gathered from five different people of varying sizes. We had each person react to a variety of annoyance stimuli. Some annoyance stimuli were acted. For example we had some people flail their arms wildly, stamp their feet, and jump backwards. Other annoyance stimuli were induced by us. For example we sometimes poked our subjects with a brook, or threw a tennis ball at the subjects to evoke a response. Each person reacted differently to the annoyance stimuli giving us a wide range of data.

We performed a greedy feature selection algorithm in conjunction with 100-fold cross validation and an SVM classifier to get offline results for our data. The classifier has an error rate of 38% when not on the robot.

In order to test the classifier on the robot, we needed to first construct a map of the environment. Since the robot was too tall to use a laser scanner to build a map, we attempted to implement a map builder using point cloud data from the Kinect.

I. INTRODU`CTION

AS robots become more and more common in society, the number of interactions between humans and robots will increase dramatically. If robots are to integrate with humans, they will need to serve their purpose without upsetting the humans it interacts with. Current personal robots already excel at avoiding humans. However, they are nowhere near as good at avoiding paths that would interfere with humans. For example, if two people are engaged in a conversation the robot may attempt to take the path in between them. Another example would be if a person was playing a video game and a robot walked in front of the screen. This paper investigates how to teach a robot that it has annoyed someone and how to avoid annoying them in the future. The first challenge we needed to overcome was a way to capture data that we could label as annoyed or not annoyed. We decided to use short video clips since individual poses and gestures would be too ambiguous in real world situations. We used the Microsoft Kinect with the OpenNI and nite frameworks to record skeleton joint positions and orientations in a short video clip. To facilitate data collection, we built a ros node the reads recorded streams of joint positions and export them in a Matlab compatible format. By extracting features from video clips we are able to classify a larger number of exaggerated reactions with less error than if we were to just use poses and gestures. Once all of the features were extracted from the video clips we used a greedy feature selection algorithm to determine what features to use in our classifier. We had over 100 features to choose from based off of joint position, velocity, acceleration, and ratios of how much the joints moved relative to the body. The results of the built in SVM classifier in MATLAB using the features from our selection algorithm is an error rate of about 38%.

II. APPROACH

Since we recorded video clips knowing which ones had annoyed people in them, it was an obvious choice to employ some form of supervised learning to teach the robot how to determine whether or not it annoyed someone.

Before we could build a classifier, we needed to determine what features we should look for. To limit the scope of the project, we only considered a limited number of annoyance scenario. We wanted the robot to be able to tell when it was too close to a person, walked between someone having a conversation, or blocked someone's line of sight. To further limit the scope of the project, we would only have our subjects act out overly exaggerated annoyance reactions.

The data we recorded included people peering around the robot when it blocked their line of sight, flailing their arms in the air, stamping their feet, and quickly jumping away from the Kinect. We felt that the best way to measure these reactions would be linear and nonlinear combinations of joint displacement, joint velocity, joint acceleration, and whether or not someone held their hands in the air for consecutive frames. However, this resulted in us having eleven features per joint. Since the Kinect tracks 15 joints,

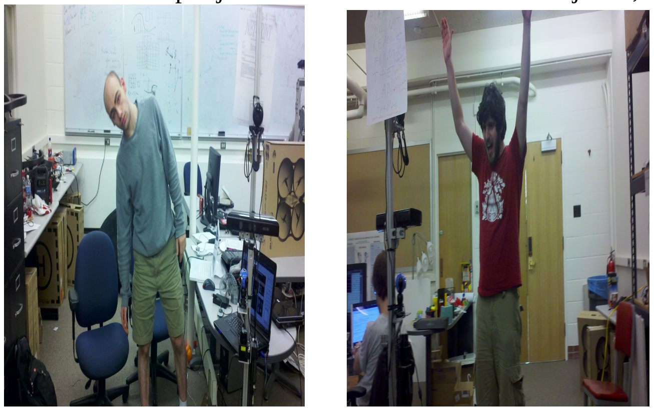


Figure 1: Examples of how people reacted when the robot annoyed them. A) The robot was blocking line of sight, so the subject had to peer around the robot. B) The robot violated the subject's personal space so the subject threw his hands in the air.

there were over 150 features that could possibly be used in our classifier. We did not have enough datasets to support that many features so we implemented a greedy feature selection algorithm with 100-fold cross validation to choose

Same Sinensky is affiliated with the Department of Computer Science, Cornell University. David Diner is affiliated with the Department of Mechanical and Aerospace Engineering, Cornell University. {scs238, dg386}@cornell.edu

the 20 best features.

The feature selection algorithm initializes two empty arrays, one for storing the features used so far and one for storing the score the of the classifier using the current set of features. The score of the classifier is defined as one minus the successful prediction rate. During each iteration, the algorithm adds the feature that results in the lowest score when running the classifiers. For each fold, the set of features that resulted in the lowest score are appended to a results array. The features we choose to use were the 20 most common features the algorithm selected during the cross validation process.

Once we obtained these features, we were able to test how well our annoyance classifier worked. We used MATLAB's built in SVM classifier and logistic regression classifier. On an offline dataset containing 250 examples, 100 training examples were selected randomly, and the remaining examples were used as test cases. The SVM classifier correctly identified 93 video clips as annoyed or not annoyed. This means our algorithm provided features that were correct 62% of the time. However, the logistic regression classifier was only able to correctly classify 82 video clips. This means that our feature selection algorithm when used with logistic regression was only correct 55% of the time. Although this result appears to be discouraging for being able to accurately predict annoyance, it is still better than randomly guessing and is a decent initial attempt at classifying something as ambiguous as human annoyance.

A. *Feature Calculation*

The Kinect published skeleton data on a per frame basis and periodically some frames or joints would not be published. To account for this, we developed a MATLAB script which took time-stamped joint data per frame and formatted the data into a matrix where each row was a time and each column was a joint variable. This large matrix was computed for each video clip we recorded. In order to use the SVM classifier in MATLAB and logistic regression, we needed to format each matrix of data into a new matrix where each row was a video clip ID and each column was a feature. Another MATLAB script was employed for this purpose. The script would use joint variable data to compute displacement, total distance traveled, average velocity, total speed, and net acceleration for each joint. The script would use these numbers to calculate what features to use for the feature selection algorithm.

We also used whether or not a person's hands were above their head for consecutive frames as a feature. If the joint variable data appears to indicate that the person was annoyed, then having their hands above their head for consecutive frames suggests that the person is staying annoyed. Conversely, if a person is not annoyed and just happens to have their hands over there head, this feature does not immediately imply that the person is annoyed.

Once all of these features were extracted from the raw Kinect skeleton data and placed into matrix form, we could use the built in SVM and logistic regression classifiers to check or results.

B. *Feature Selection Results*

The results of our feature selection algorithm were not too surprising. The most popular features were knee and joint velocity and acceleration. This makes sense because in almost all of the annoyance reactions, there was a sudden change in these joint's position. Although the change in position might be comparable to when someone is just going about their daily business, the fact that they suddenly change is significant. Our feature selector was able to capture this real world observation. Additionally, joints tend to move quicker when a person is annoyed versus when they are simply milling about. The feature selection algorithm was able to capture this, but only for knees and elbows. We had thought that the selection algorithm would choose to use the velocities of hands and feet since they were both moving a lot when our subjects were flailing there arms and stamping their feet. Also, the selection algorithm found that whether or not a person had their hands above their head for consecutive frames was a significant detail.

The one surprise from our feature selection algorithm was that ratios of joint velocities and accelerations to the body velocity and acceleration were almost never used. We had thought that when a person was stamping their feet, peering around an obstacle, or flailing their hands that the person's joints would be moving and that the body would be mostly stationary. This would mean that for annoyed reactions this ration would be very large. However, if a person was going about their daily business we were expecting a ration of approximately one. We had thought that this would allow for the classifier to make distinctions between annoyed and not annoyed. However, this was not the case. See the appendix for a table summarizing the results of our feature selection algorithm.

C. *Offline Results*

The results of our feature selection algorithm are summarized in the figure below. Additionally, the results of testing the training data against itself, the cross validation tests, and final offline testing are summarized below. The average error for testing the training data against itself was 1% for the SVM classifier and a maximum error of 9.7%. The logistic regression classifier had a surprisingly large average error of about 24% with a maximum error of 30% when testing the training data against itself. While performing the 100-fold cross validation, the SVM classifier

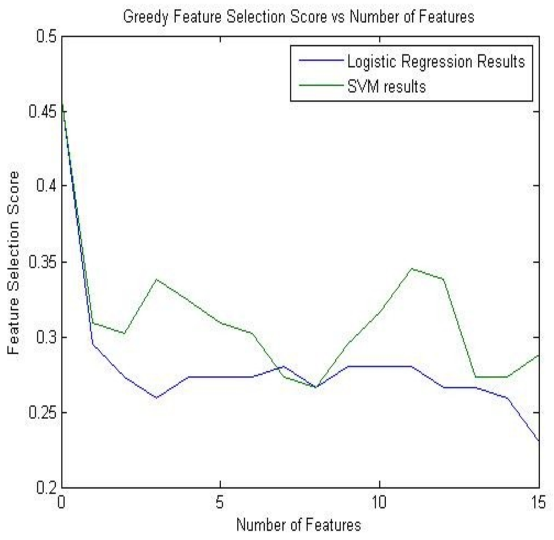
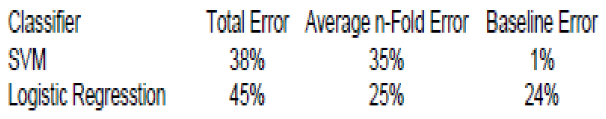


Figure 1: Offline Results while running 100-fold cross validation



Classifier	Total Error	Average n-Fold Error	Baseline Error
SVM	38%	35%	1%
Logistic Regresstion	45%	25%	24%

Table 1: Results from two different classifiers

had an average error of about 35%, whereas the logistic regression classifier had an average error of about 25%. The final offline testing results using random test and training data out of the 250 datasets we collected are: 38% error for SVM classification and 45% error for logistic regression classification.

III. EXPERIMENTS

We initially planned to have robot navigate though a room using a cost-map that was augmented with additional costs to avoid annoying people. However, we were unable to get this working on the robot.

IV. NAVIGATION ISSUES

We initially planned to build a static map of the room using SLAM to seed the cost-map. However we encountered some issues. We eventually got 2D slam working with a pointcloud_to_laserscan simulator, but the original version missed many of the obstacles in the map.

Next we tried using rgbdslam, but it was too resource intensive and when we projected the maps down, we still needed further processing for the map.

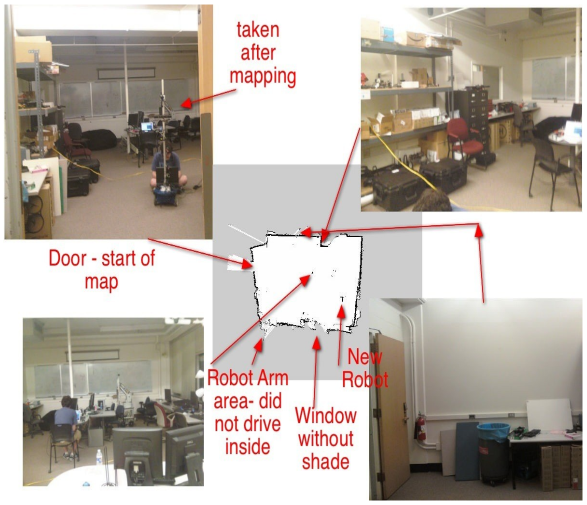


Figure 2: Map of the Robot Learning lab using the built in gmapping package in ROS. Although it is able to detect the walls, tables, chairs, and other objects in the lab are not marked as obstacles

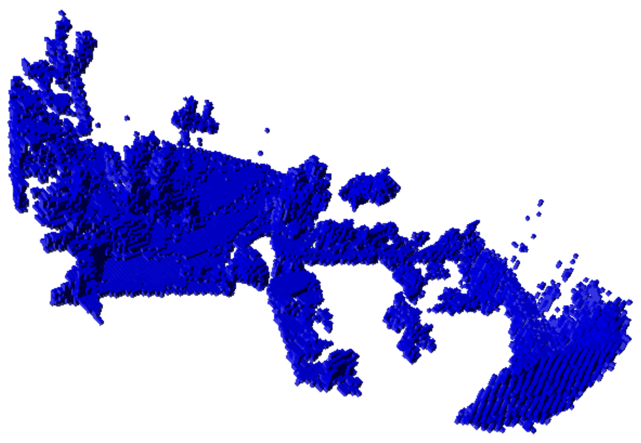


Figure 3: A 3D Point Cloud Representation of the Robot Learning Lab

Our third attempt to build a map with the Kinect point cloud data was to use the point cloud library to filter out the floor, and then use the largest depth in each column of the point cloud to convert the Kinect point cloud to a fake laser scanner. This method would require using the built in RANSAC planar filtering algorithm in the point cloud library, then removing them from the Kinect's point cloud data. We would then train the robot using a floor classifier so that it would

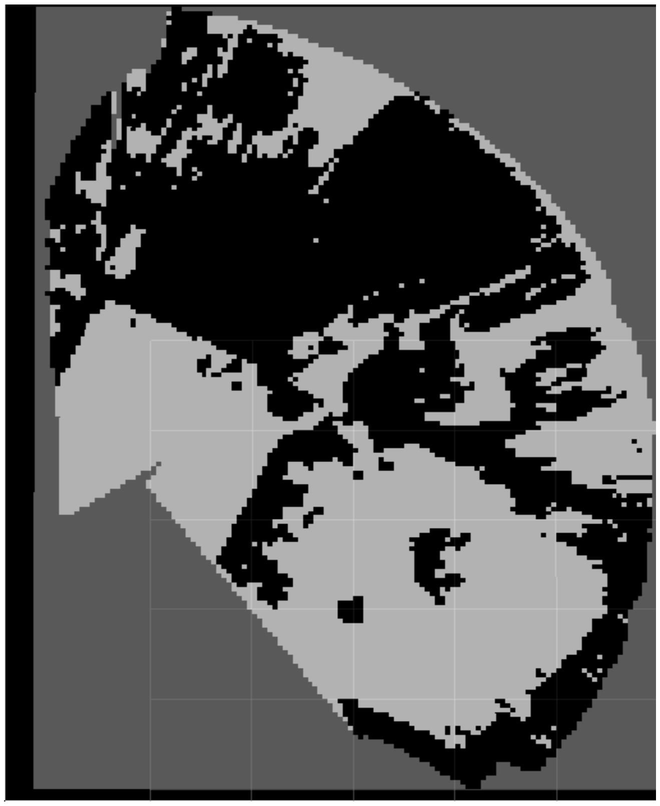


Figure 4: 2D projection of the RGBDLAM map of the Robot Learning Lab Note that the part of the lab with the robotic arm is completely marked as an obstacle

not use the depth data from points corresponding to the floor when converting the point cloud data to laser scan data. Although we were successfully able to filter images in offline experiments, the time required to filter each point cloud was prohibitive to using it on the actual robot.

It turned out there was simpler solution. We modified pointcloud_to_laserscan to take the closest reading at any height range that did not include the floor. We put a lower limit on the height by measuring the configuration of the Kinect on the robot. This worked well to produce maps the showed most obstacles, but the maps were slightly noisy.

Since we were unable to get the navigation stack working, it was necessary for us to use simple control commands when performing the experiments on the robot.

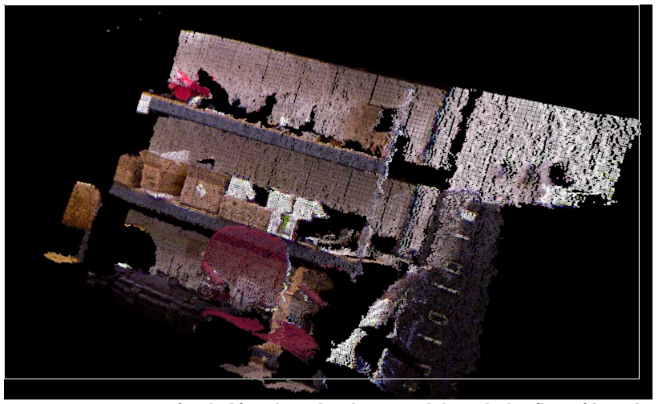


Figure 5: Picture of a shelf in the robot learning lab with the floor filtered out

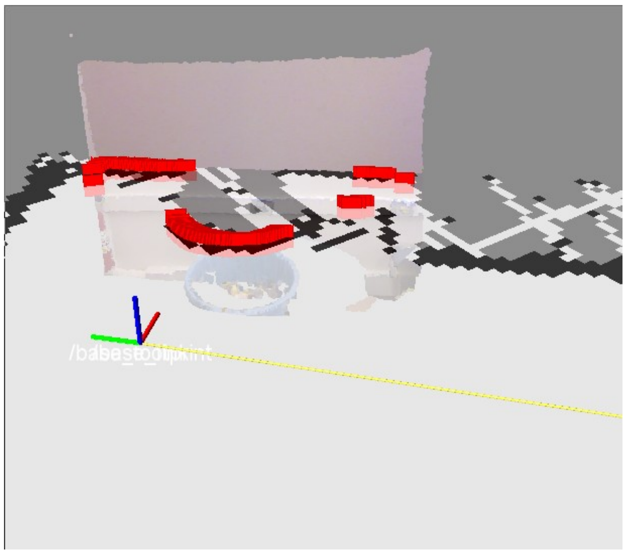


Figure 6: Map of the corner of the Robot Learning Lab with a garbage can and table. Even though the garbage can and table are not in the same plane as the laser scans, they are still added as obstacles to the map.

Our solution was to have the robot drive in the direction of a person it detects. If the robot detects that it annoyed the person it will stop and back up. If it does not detect that it has annoyed the person, it will attempt to follow the person. We were unable to get this setup working either. Although this was not how we intended to experiment with the robot we were still able to test our annoyance detection algorithm in situations similar to those a robot would experience.

V. FUTURE WORK

A. *Finding Better Features*

Although we were satisfied with the features our classifier could select from, better choices must exist that would reduce the error of our classifier. The robot would need to know what context of annoyed poses in order to consistently and accurately identify human annoyance. These context clues can probably be implemented as features and increase the ability of the robot to learn when it is annoying humans.

For this project, we limited ourselves to Kinect skeleton data. However, body posture is not the only way the humans convey that they are irritated. Audio cues could be employed to increase the number of significant features the classifier can use. For example someone who is annoyed may make a sudden loud noise or make a fake coughing noise. Since the Kinect also has a microphone array, it should be able to process these inputs to provide a wider and hopefully more useful range of features to use.

B. *Using Hidden Markov Models*

When we actively run the classifier on the robot, we use the five most recent frames as test data to feed into the SVM classifier. While doing offline testing we noticed that people who were annoyed in one frame, were very likely to still be annoyed in the next. Similarly, people who were not annoyed in one frame were very like to remain not annoyed in the next frame. Using this observation, we could create an HMM which would encourage the robot to think a person remained in their previous state unless the sensor data told them otherwise. Although this would not help with the initial classification of whether or not a person was annoyed, this would produce smoother results while running on the robot. This means the robot is less likely to jump between states which would also help reduce how much the robot is annoying the person.

Although an HMM would help to reduce how much a robot annoys a person, it would be difficult to obtain the transition probabilities since the offline testing is not done on a per frame basis. This means that in order for us to build an HMM we would need to know exactly when a person transitioned from annoyed to not annoyed and vice versa. This is not easy since we would have to synchronize our real time results with the frames that our skeleton capture program records. However, if we could find a way to do this, the results on the actual robot would definitely improve.

C. Reinforcement Learning

Although we built an annoyance detector with moderate success, we were unable to teach the robot how to navigate through a room without annoying people. This would require two things from the project. First, the classifier would need to have higher accuracy in order for the reinforcement learning algorithm to have a chance at working. Second it would require the robot to be able to remember objects of interest and poses humans make when they are in an annoyable state.

To perform the reinforcement learning we would also need a way of storing and learning from joint orientations. This would allow the robot to learn when a person is facing an object of interest as well as when two people are facing each other. If the robot is able to learn that whenever it walks through two people having a conversation that it annoys them, it could use reinforcement learning to avoid paths that go between people who are oriented towards each other.

Similarly, the point cloud data from the Kinect could be used to store objects of interest. If the robot walks between a person and a painting on a wall, we could use a combination of reinforcement learning and supervised learning to teach the robot what certain objects of interest look like. Supervised learning would give the robot some initial objects of interest and reinforcement learning would be used when the robot walks between a person and an unknown object of interest. If the robot detects that he annoyed someone by blocking their line of sight to an object of interest, the robot will learn not to walk between a person oriented towards an object of similar shape.

Although our classifier for detecting annoyance is a small step towards teaching robots to avoid annoying humans, there are still many areas that need to be explored more thoroughly before personal robots can smoothly interact with humans in a human environment. In addition to needing better features, our project also indicated that a reliable mapping and localization implementation needs to be made for taller robots. Although there are tools to convert a 3D map to a 2D occupancy grid, creating the 3D map is time intensive and the implementation we found on ROS was prone to crashing. However, since the robot is equipped with a Kinect, it should be possible to generate an accurate map of an environment regardless of the robots dimensions.

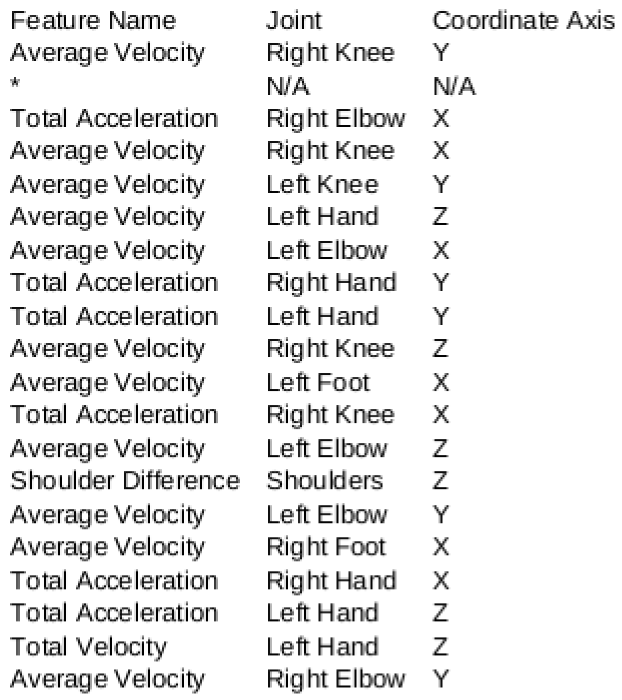
VI. Conclusion

To summarize, the goal of our project was to teach a robot to detect whether or not it was annoying a human. To achieve this goal, we used skeleton data from the Kinect to extract significant features to use in an SVM classifier. Although we had a moderate offline success rate of 62%, we were not able to thoroughly test the annoyance detector on the robot.

Looking at the rising demand for person robots to do menial labor around the house, it becomes apparent that robots need not only navigate through physical space, but they must do so without disturbing the user. If in the process of doing its task, the robot upsets its user, then the robot is not doing its job correctly. The ultimate goal of this line of study is to teach robots how to navigate through human populated areas without irritating any humans.

Appendix

Results from forward selection algorithm



Feature Name	Joint	Coordinate Axis
Average Velocity	Right Knee	Y
*	N/A	N/A
Total Acceleration	Right Elbow	X
Average Velocity	Right Knee	X
Average Velocity	Left Knee	Y
Average Velocity	Left Hand	Z
Average Velocity	Left Elbow	X
Total Acceleration	Right Hand	Y
Total Acceleration	Left Hand	Y
Average Velocity	Right Knee	Z
Average Velocity	Left Foot	X
Total Acceleration	Right Knee	X
Average Velocity	Left Elbow	Z
Shoulder Difference	Shoulders	Z
Average Velocity	Left Elbow	Y
Average Velocity	Right Foot	X
Total Acceleration	Right Hand	X
Total Acceleration	Left Hand	Z
Total Velocity	Left Hand	Z
Average Velocity	Right Elbow	Y

Table 2: The 20 features our selection algorithm predicted.
* Is the Boolean value for whether or not the person's hands were in the fair for more than three consecutive frames during the video clip.

Acknowledgment

-Professor Ashutosh Saxena
-Colin Ponce
-PCL
-OpenNI
-ROS

References

[1] R. B. Rusu and S. Cousins, “3D is here: Point Cloud Libarary (PCL)”, *IEEE International Conference on Robotics and Automation*, May 2011.

[2] K. M. Wurm, A. Hornung, M. Bennewitz, C. Stachniss, and W. Bugard, OctoMap: A Probabilistic Flexible, and Compact 3D Map Representation for Robotic Systems, *Proceedings of the ICRA 2010 Workshop on Best Practice in 3D Perception and Modeling for Mobile Manipulation*, May 2010.