Robot Autonomy Final Report : Team Husky

Amit Bansal
Master student in Robotics Institute, CMU
ab1@andrew.cmu.edu
Akshay Hinduja
Master student in Mechanical Engineering, CMU
ahinduja@andrew.cmu.edu
Zhichao Jin
Master student in Robotics Institute, CMU
zhichaoj@andrew.cmu.edu
Yuzhang Liu
Master student in Mechanical Engineering, CMU
yuzhangl@andrew.cmu.edu
Akash Anandrao Sambrekar
Master student in Mechanical Engineering, CMU
asambrek@andrew.cmu.edu
Nai-Wei Su
Master student in Mechanical Engineering, CMU
naiweis@andrew.cmu.edu

Abstract

This report focuses on the human detection part of the HUSKY project. Pointcloud data is collected by Velodyne LIDAR and then processed in both C++ and MATLAB. The two approaches help us to detect the human accurately. Both approaches focus on clustering and region of interest detection of the pointcloud. In C++ approach we applied Euclidean and Region Growing clustering, while in MATLAB approach we applied DBSCAN clustering and machine learning to detect human. By now we are able to detect multiple moving humans in real time with high accuracy.

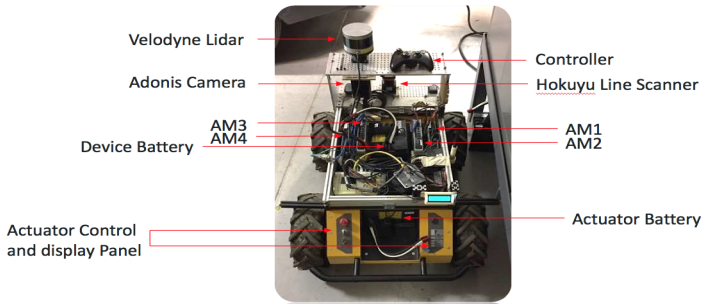

Figure 1: HUSKY Robot

1 Problem Definition

The main theme of the project is the development of intelligent robotic systems that can work with humans in a team. To support complex cognitive capabilities, various technologies from multidisci-

Submitted to 30th Conference on Neural Information Processing Systems (NIPS 2016). Do not distribute.

plinary fields have been leveraged including high-level reasoning, natural language understanding and semantic perception. A great example of the application of this robot is the in battlefield. When the soldier needs to detect if there's any potential threats or enemy in the nearby environment such as a building or a house, he or she can send our robot into it by voice command and inspect the building. To achieve this task, a significant ability is to know where objects or the enemy, or human, is. However, for a human interactive mobile robot, it is more likely for the robot to hit a human than a static object. Thus, the significant ability about detecting, recognizing, and tracking a human being should be built in this robot. This project is mainly about analyzing the point cloud data collecting from a Velodyne LIDAR to achieve detecting, recognizing, and tracking a walking human.

2 Related Works

The commonly used approaches for detection and tracking of moving objects for vehicular applications involve sensors such as radar and LIDAR. They are capable of collecting data directly. In particular, recent models of laser scanners are capable of gathering high-resolution data at high scanning speeds.

One of the work using LIDAR data related to us is Luis et al. describe the application of pedestrian detection and tracking using LIDAR Data. The approach in their method is first quickly select human potential points in the point cloud LIDAR collected, then use statistical pattern recognition techniques to classify each object. The algorithm uses geometric and motion features to recognize human signatures. The main improvement is most significant for static human.

Another work related to us is Gabriel J et al. method utilize structure model to recognize human and object from motion 3D point cloud. They propose an algorithm for semantic segmentation based on 3D point clouds. They introduce features that project the 3D cues back to the 2D image plane while modeling spatial layout and context. A randomized decision forest combines many such features to achieve a coherent 2D segmentation and recognize the object categories present.

In 2008 Thornton et al. report an algorithm capable of detecting both stationary and moving humans. The paper gives an approach for the automatic detection and tracking of humans using multi-sensor including 3D Ladar and long wave infrared video and integrated the data from these sensors.

3 Approach

3.1 Data Collection Procedure with the robot

The Velodyne LIDAR is mounted on the front of the robot and connects to the third computer, AM3, on the robot. There are several procedures towards successfully collecting the point cloud data. First, we established the connection between our local computer and AM3. Then we ran the ROS package on AM3 to start collecting data. While AM3 is collecting data, we can visualize the point cloud on our local machine. In this process, some adjustments need to be done to account for the instability of the hardware system. After finishing data collection, we mounted our USB drive on to AM3 and copied the bag file to USB before inspecting the bag file. Finally, we re-visualized the data in RVIZ on local machine to make sure the bag file is intact and the file is ready for analyzing.

One of the problems that we encountered was, if we record the ROS bag locally into our machine (by subscribing to the ROS Master) , there would be huge communication lag and almost 30 percent of the data would be lost.

3.2 Clustering and Region of Interest Detection

Cluster analysis or clustering is the task of grouping a set of objects in such a way that objects in the same group are more similar to each other than to those in other groups.Similarities can be found through attributes such as distance, curvatures, colors and several others. To get real time results we used C++ for clustering and Region of Interest Detector, and subsequently fused it with the Kalman Filter Tracking algorithm. However, to improve human detection, we trained a fully connected layer with 400 hidden units in MATLAB. The approach for both these methods are mentioned below :

3.2.1 C++ Approach

We have tried two methods: Eucledian Clustering, and Region Growing Clustering. We will explain both briefly as follows:

3.2.1.1 Euclidean Clustering:

A simple data clustering approach in an Euclidean sense can be implemented by making use of a 3D grid subdivision of the space using fixed width boxes, or more generally, an octree data structure. This particular representation is very fast to build and is useful for situations where either a volumetric representation of the occupied space is needed, or the data in each resultant 3D box (or octree leaf) can be approximated with a different structure. In a more general sense however, we can make use of nearest neighbors and implement a clustering technique that is essentially similar to a flood fill algorithm.

3.2.1.2 Region Growing Clustering:

The natural extension of the Euclidean clustering algorithm for the problem of segmenting points with similar properties together, is to include additional information in the checks performed at step 3, such as the point's color, or an information regarding its surrounding geometry, etc.This method is similar to a region growing approach, with the difference that it propagates the growth along directions of lower curvature first. Points will stop being added to the current region if none of their neighbors fulfills the angle criteria and the queue of points is empty.

After trials and estimating which gave the most consistent and dense clusters, we went ahead with the region based clustering algorithm. We use an Oct-Tree approach where in we save the first pointcloud into the Oct-Tree. As the sensor stream feeds in, we use this Oct-Tree to remove static points from the incoming poinctloud. This is called background subtraction. Then on applying the Region Growing Clustering, we get clusters of moving human like objects, based on the parameters we enter for the curvature, size etc. These cluster indices are then passed on to the tracker.

3.2.2 MATLAB approach

We applied another approach to cluster the pointcloud and detect the region of interest. The work flow chart is shown in Figure 2. The major steps are: Ground Removal and DBSCAN Clustering, XZ-Plane Projection and Re-sizing, and Machine Learning. This approach is implemented in MATLAB.

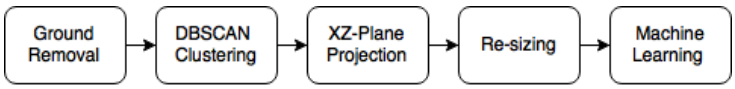

Figure 2: Work Flow

3.2.2.1 Ground Removal and DBSCAN Clustering

After the raw pointcloud data is collected, it is first downsampled for processing in the next steps. To remove the ground, our approach is to fit a plane of the ground based on the maximum distance, the reference vector ([0,0,1], pointing upward) and maximum angular distance. The points are then classified into inliers and outliers, where inlier are the points that fits the ground plane and outliers are the other points. To remove the ground, the inlier points are removed.
The clustering is achieved by applying DBSCAN, which stands for Density-based spatial clustering of application with noise. It clusters the pointcloud based on the density of the points and returns outliers and inliers.
The ground removal and DBSCAN clustering process is shown in Figure 3. The ground is labeled by green square and the human is labeled by red square.

3.2.2.2 XZ-Plane Projection and Re-sizing

The clustering of the pointcloud gives several potential human cluster of the points. To further detect if the cluster is human or not, the point cluster is first projected onto X-Y plane and check if the size of the projection is within range of human projection. If the cluster passes this test, it is then projected onto XZ-plane to generate a picture for the training part.

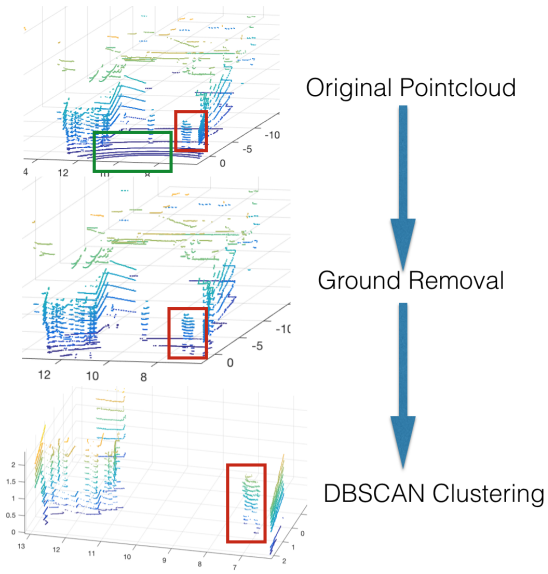

Figure 3: Ground Removal and DBSCAN Clustering

However, the projected graph has size of 170*100, which gives 17000 features for the learning process for each graph. To shorten the process, the picture is then downsampled and re-sized to 52*30, which only gives 1560 features for each graph.
The XZ plane projection and re-sizing process is shown in Figure 4. The non-human cluster is shown in left and the human cluster is shown on right.

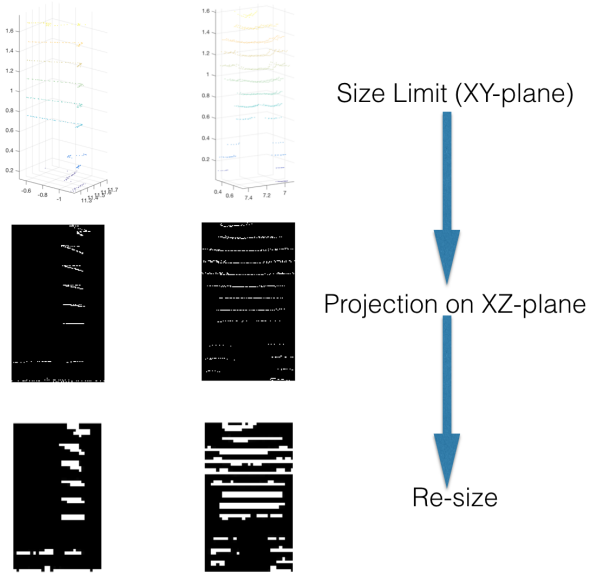

Figure 4: XZ-Plane Projection and Re-sizing

3.2.2.3 Machine Learning

To achieve better accuracy in Object Detection , we decided to use a Fully connected layer with 400 hidden layers for the training. The features that we expected the algorithm to learn are as follows :

1. Density of Point Cloud (Heat Map Belly Dense)
2. Symmetrical Distribution of points

3.2.2.3.1 Data Labelling

Since , there is no standard data-set available for training on Velodyne LIDAR point cloud images , we had to collect and label our own data. We labelled the data ourselves and generated 3000 images for running our classifier. Here is the logic for the labelling :

Human: label 1
Nonhuman: label 2
Not sure: label 3 (exclude from data later)

3.2.2.3.2 Architecture

Our architecture has a fully connected layer with Backward propagation. We use Sigmoid function to output the probabilities of the class and label the data with the highest probability.

Cost Function : Sum of Log probabilities of Correct Label

Input size = 52*30
Output Classes = 2
400 hidden units
Train Data = 3000 images
Test Data = 200 images
2 Validation Data Sets : 80 images

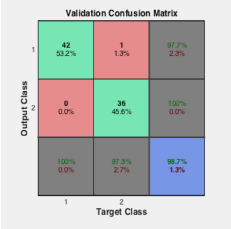

Figure 5: Validation Test Result

3.3 Tracking

After noise removal and ground plane segmentation, the point cloud is clustered into Regions of Interest. Each of the cluster is assigned an Object Identifier number . The object Identifier number stores the cluster center for the cluster recognition. The tracker uses the Centroid of the Cluster for re-assigning the Object Identification Number from one frame to the next.

The tracking problem comprises of two main steps :

3.3.1 Data Association

The data association is done by using the minimum Euclidean Distance Algorithm. Let Object Identifier y be associated with cluster with centre x, at a given frame. In the next frame, the cluster whose center is closest (by Euclidean Distance) to x would be associated with the object Identifier y. This process is repeated for all the object identifiers to get the cluster association from one frame to the next.

3.3.2 State Estimation and Prediction using Kalman-Filter algorithm

The 3-D point cloud is projected down to the ground plane and the tracking is performed on the cluster centre. The state dimension for the tracking is 4 i.e (position in x , position in y , velocity in x , velocity in y). The measurement matrix consists of 2 Dimension i.e Velocity in x and velocity in y .

There is no control input for our problem statement. Action Uncertainty (Q) = 0.01; Measurement Noise (R): sigma=0.1;

In all the Tracking algorithm follows the sequence :

Figure 6: Tracking Workflow

4 Results

We were successfully able to detect and track Humans, even with a noisy sensor data. The Region-Growth based Clustering algorithm gives us precise results for identification of Humans. We combine that with the Kalman-Filter algorithm to achieve reliable tracking of humans.

Here are some of the results from each step of the Perception Algorithm :

4.1 Data Collection :

Figure 7, depicts the raw data (noisy) collected from the Velodyne 16 Channel LIDAR on the Husky. The data is huge and needs to be downsampled before implementing clustering on it.

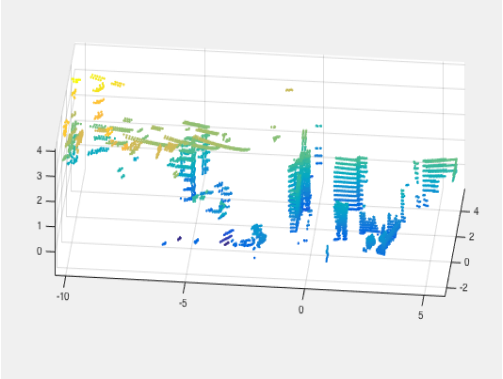

Figure 7: Raw data from Velodyne LIDAR

4.2 Clustering and Region Of Interest Detection :

We implemented two algorithms for cluster detection : Region Growth Clustering and DBSCAN based clustering. The results are mentioned in the figures below :

4.2.1 Region Growth Clustering Result

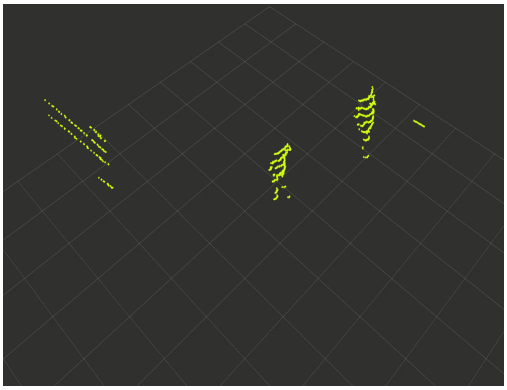

Figure 8: Region Growth clustering result

4.2.2 Clustering using DBSCAN results

DBSCAN algorithm is based on Eucledian Distance. If a point is withing a certain radius of the cluster center, it is grouped to that particular cluster . However, if the number of points in the cluster (formed after grouping all points) is less than the threshold value, the cluster is deemed noisy and is assigned a value of 0.

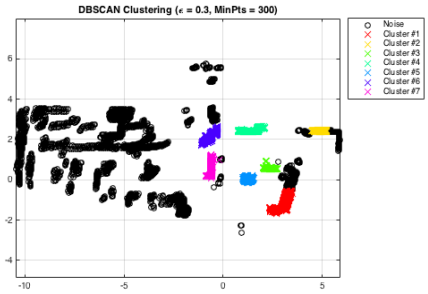

Figure 9: DBSCAN clustering result

4.3 Tracking :

The tracking uses the Kalman Filter algorithm described in the previous section. The figure below depicts four people walking around in front of the robot. The colored boxes represent each of the human with the exception of the black box (which represents centre of the frame) and the blue box which represents a wall.

As the Human moves, the colored boxes slide with them. The color of the box associated with a particular human does not change from one frame to the next, because we used Data Association in Kalman Filter.

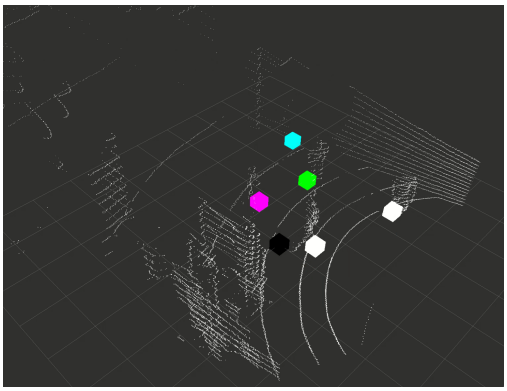

Figure 10: Tracking

The video link mentioned below demonstrates all our subsystem capabilities.

5 Work Distribution

Based on the final aim of the project, we have created different work modules in the progress. The distribution of work is done in accordance to these modules and listed below:

Work Distribution	
Work Modules	**Responsible group members**
Data Collection from Husky's Sensors	Entire team. All team members need to know how to operate the robot and extract data from its sensor
Point cloud clustering	Amit and Akshay
Recognition of segmented point-clouds	Akash and Nai-Wei
Tracking of multiple humans	Yuzhang and Zhichao
Verification and analysis	Entire Team. All the team members need to contribute in validation of individual units of code developed and integration of the same under one package

: Table 1: Work distribution table

6 Link to Video

Links to the videos that show our real-robot demo are listed below:

6.1 Data Collection

The robot is static and the data is collected from a 16 channel Velodyne LIDAR.

`https://youtu.be/QUat9r952qc`

6.2 Clustering Algorithm

`https://youtu.be/lEGH4WYtu4Y`

6.3 Real-Time Tracking Algorithm

a. With two people walking

`https://youtu.be/s-2Ef98TA3E`

b. With four people walking

`https://youtu.be/FBUwrIYg_VE`

7 References

[1] Munder, Stefan, Christoph Schnorr, and Dariu M. Gavrila. "Pedestrian detection and tracking using a mixture of view-based shape–texture models." IEEE Transactions on Intelligent Transportation Systems 9.2 (2008): 333-343.

[2] Brostow, Gabriel, et al. "Segmentation and recognition using structure from motion point clouds." Computer Vision–ECCV 2008 (2008): 44-57.

[3] Zhong, Yu. "Intrinsic shape signatures: A shape descriptor for 3d object recognition." Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on. IEEE, 2009.

[4] Thornton, Susan M., Mike Hoffelder, and Daniel D. Morris. "Multi-sensor detection and tracking of humans for safe operations with unmanned ground vehicles." Proceedings 1st. Workshop on Human Detection from Mobile Robot Platforms, IEEE ICRA. IEEE. 2008.

[5] Oh, Jean H., et al. "Toward Mobile Robots Reasoning Like Humans." AAAI. 2015.