Cognitive robotics(MA-INF 4304 SS2016)

Lab/Congnitive Robotics - Laser-based Perception of Small Objects on the Ground

Rania Briq, Advisor: David Droeschel, Supervisor: Professor Sven Behnke

Universität Bonn
rania.briq@outlook.de

Abstract. This report describes our work on the challenge "Laser-based perception of small objects on the ground". The description of this challenge appears in The Mohamed Bin Zayed International Robotics Challenge(MBZIRC), an international robotics competition. The task objective is to have a set of UAVs collaborate together for the purpose of searching for, locating and tracking both static and moving objects on the ground so that they may picked up and placed down. The task is associated with many challenges in the field of robotics and has several practical applications. We will discuss how we tackled the problem, describe the algorithm that we designed and analyze its results.

1 Introduction

Detection of small objects on the ground from laser scans is a challenging task. The difficulty first lies in the ability to discern the object on the ground. Secondly, both the objects and the UAV scanner are in motion, the terrain is uneven and UAV is free to rotate in three dimensions in dynamics such as roll-and-pitch. Our task consists of different stages, each of which deals with a certain subtask. The first is Obstacle detection which is targeted at identifying the set of of points that make up an obstacle. The second subtask pertains to clustering the obstacle points into the object that they belong to. The third subtask addresses tracking these detected objects. We test our algorithm on images that were captured from 2 different scanners, one is in motion but with stationary obstacles, and the other is stationary but with moving small and big obstacles. In the end we give a visual analysis of the results.

2 The Setting

The type of UAVS that we used in this lab is Velodyne Lidars that are mounted with 16 laser diodes and capture real-time 3D scans. Laser scanners operate based on a simple principle: they measure the distance from an object transmitting a light pulse to the target area, measuring the time that it takes for the beam to hit the object and for the reflection to come back. The advantages of having multiple scanners the UAV can roll sideways and therefore in the absence

of multiple scanners it could possibly miss objects on the ground. Having scanners placed at several angles reduces the probability of missing the object and improves the visibility range. The second advantage is that it helps in forming a detailed view of the detected object. The scanner captures the data in a circular manner and the scans appear as rings where the further away the scanner is, the bigger the radius of the ring and vise versa. The measurements gathered by the scanner are represented in a Point Cloud object and contain information such as data points coordinates (The origin is the Lidar's coordinate), color intensity and ring number.

Below is a raw image data captured by the scanner color-coded by the color intensity. Each ring corresponds to a scanner number.

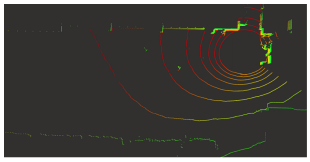


Fig. 1. Laser scans

Fig. 2. A stationary Lidar scanner

Fig. 3. The experiment environment captured during scan time. The moving objects were small remote control cars representing the small obstacles and people representing the big obstacles

3 Algorithm

3.1 Overview

As input, we have a raw 3D set of scan points. The question is, how do we utilize the measurements that were captured by the scanner in order to accomplish our task. To that end, we designed a pipeline that consists of 3 recurrent stages, each solving a subtask of the general problem. The first stage aims at detecting the individual obstacle points. It takes a set of points as its input and tries to identify the obstacle points among them. The output can be thought of as an unordered set of points that need to be identified as a set of individual objects. We use clustering to form these objects and this comprises the second stage in the pipeline. The last stage solves the tracking problem; a moving object needs to be identified and clustered in a consistent manner from scan to scan, this entails assigning the same cluster id to a moving object thereby ensuring that the id remains fixed over time.

3.2 Obstacle Detection

Obstacle detection in laser scans can be carried out using different approaches with varying degrees of robustness [Burgard, 2009, Montemerlo et al., 2008]. The first and simplest approach that we experimented with relies on estimating the height difference from the ground and comparing them with neighboring measurements. The method requires determining a grid size that indicates the neighboring area based on the (x,y) component and then inspect the difference in the z component (height from the ground). The grid is then slided over the whole set of points in the scan. For each such placement, we find the minimum z value in the grid and check which points inside the cell exceed this value by more than a given threshold. This method requires determining a grid size and a threshold value. Too small a size could result in false negatives seeing that the

probability for the difference between the neighboring points to be so large that it exceeds the threshold is small. On the other hand, a large grid size could result in false positives seeing that the neighboring area is too big and the probability is higher that non-obstacle points will be detected as obstacles if the threshold is not set right. In addition to that, this method is susceptible to noise and discrepancies that occurred as a result of an uneven terrain or the fact that the distance to the ring is affected by the angle of the scanner. This means that the change in distance is not actually due to an obstacle, but rather due to a change in the angle or the height of the scanner. Moreover, the method does not take into account the angle of the scanner which also affects the height difference.

A more robust approach relies on estimating the distance between the rings. The assumption is that when there's no obstacle, the distance remains fixed and that obstacle points push the circles closer together thereby constituting a discontinuity in the distance between the rings that can be used for the detection. This approach is robust against the pitch-and-roll dynamics of the UAV such that even if within the same ring the continuity breaks, the distance between the rings does remain fixed. To perform the detection, we determine a window size and take measurements inside some window across two consecutive rings hoping that this area is not actually an obstacle. We then calculate the median of these points inside and take it as the approximate value between two rings in the absence of obstacles on the selected window. This constitutes the initialization step of the approach. To determine the obstacle points, we compare each point in the scan to this value, and if we find that it exceeds a certain threshold then we label it as an obstacle point. In this approach, a problem could arise in which the median was determined by obstacle points because our window fell over a big obstacle. To resolve that, one could place several windows between different rings rather than only two consecutive rings and determine the median by the majority.

The third approach relies on slope estimation. The method takes into account the angular difference between the laser diodes and estimates the distance to the ring based on the angular difference. This method still requires estimating the height from the ground, which in our case was the height of the stationary Lidar Scanner. In the illustration below, r_0 denotes the distance to the ring, h the height of the UAV and α_0 ($\beta_0 = 90 - \alpha_0$) the angle of the UAV. Once we have an estimation of the height, r_0 is given by the simple trigonometric formula

$$r_0 = \frac{h}{\sin(\beta_0)}$$

The distance estimation between the rings is then given by:

$$r_\Delta = r_0 - r_1 = r_0 - \frac{h}{sin(\beta_0 + \Delta_{01})}$$

where Δ_{01} is the difference between the angles of the two beams. We then use the first scan for initializing the expected value of the difference, and if the distance deviates from the expected distance by more than some threshold then

we mark it as an obstacle. In our case, the angle δ_{01} is 2 degrees as specified in the specification of the Velodyne Lidar scanner.

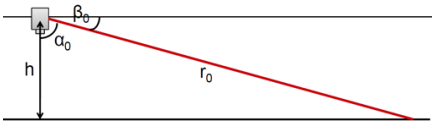


Fig. 4. Slope estimation [Burgard, 2009]

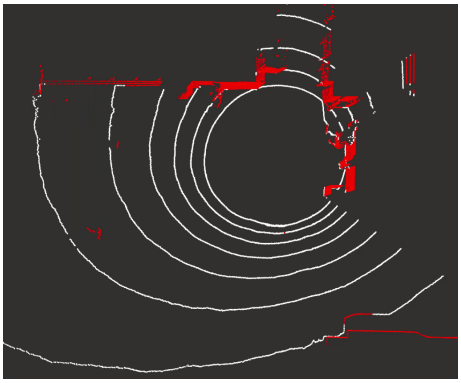


Fig. 5. The points that are marked red are the obstacle points and the white points on the rings belong to the ground. The big obstacles on the boundary belong to the walls

3.3 Clustering

The input to this component is a set of individual stand-alone obstacle points. Clustering in this stage is responsible for aggregating these points into the objects that they belong to. The output of this stage is a set of clusters which correspond to the obstacle objects. We don't rely on the order in which the points are transmitted at all, since our experiments have shown that in a ROS environment, the order in which the data points are transmitted is not necessarily the order in which they occur in the scan. Naturally, if we could make such an assumption about the order, then clustering would have been much simpler. In clustering, we need an algorithm that can identify the number of clusters k rather than one that requires k as an input parameter. To that end, we utilize Single-Link Hierarchical Clustering. Below is a template of the algorithm:

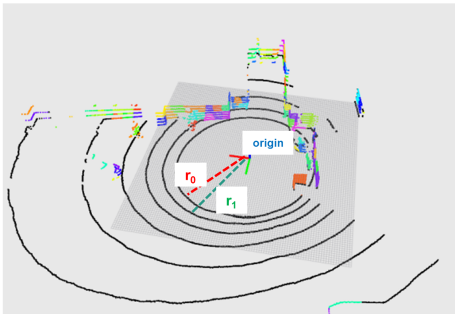


Fig. 6. Description of the method

1. *Begin with the disjoint clustering having level L(0) = 0 and sequence number m = 0.*
2. *Find the least dissimilar pair of clusters in the current clustering, say pair (r), (s), according to*

   d[(r),(s)] = min d[(i),(j)]

   where the minimum is over all pairs of clusters in the current clustering.
3. *Increment the sequence number : m = m +1. Merge clusters (r) and (s) into a single cluster to form the next clustering m. Set the level of this clustering to*

   L(m) = d[(r),(s)]
4. *Update the proximity matrix, D, by deleting the rows and columns corresponding to clusters (r) and (s) and adding a row and column corresponding to the newly formed cluster. The proximity between the new cluster, denoted (r,s) and old cluster (k) is defined in this way:*

   d[(k), (r,s)] = min d[(k),(r)], d[(k),(s)]
5. *If all objects are in one cluster, stop. Else, go to step 2.*

Fig. 7. Hierarchical Clustering Algorithm. We maintain a square matrix that represents the similarity for each 2 clusters(i,j). As clusters get merged while relying on the similarity matrix, we update its values accordingly.

In the initialization step, each point is assigned to its own cluster thereby forming a disjoint set of clusters of size n where n is the number of points. In each iteration of the algorithm main loop, two clusters are merged based on a similarity measure that is inferred from some predefined metric. The process is repeated until a predetermined stopping criterion is met. In each iteration, we merge the two clusters with the minimum distance among all pairs of clusters. By the single link criteria, the minimum distance between two clusters is defined to be the minimum distance between any pair of points $(x_i, x_j)$from $cluster_i$ and $cluster_j$. A different linkage criteria is the mean average distance linkage which is the sum of distances of all pairs from the 2 clusters, i.e.

$$\frac{1}{|C_i||C_j|}\sum_{x\in C_i}\sum_{y\in C_j} d(x,y)$$

. We have experimented with this criteria but it proved too time-consuming for our real-time purposes due to the quadratic complexity for the changes in the distance matrix at each merge. A single-link criteria allows us to perform the change and find the minimum distance linearly due to the increasing monotonicity nature of the distance as a result of merging two clusters. As a stopping criterion, we stop the merge loop when all clusters have become too far apart to merge [Goldberger and Tassa, 2008]. Two clusters are considered too far apart if the minimum distance between points in those two clusters is large in comparison to the internal distances in at least one of these clusters, which would have increased the variance of the merged cluster by far therefore possibly merging two obstacles. We define the internal distance of a cluster C of n points to be

$$dist_{int} = \frac{n(n-1)}{2}\sum_{i\neq j, x_i, x_j \in C} d(x_i, x_j)$$

At the end of this stage, we have identified a set of objects that make up the obstacles on the ground from a single scan and their number. We only use this algorithm for the first scan to get an estimate for the number of obstacle objects. We then take the hypothesis of these clusters and feed them to the next stage in the pipeline. Our cluster hypothesis consists of the center point for each cluster, and the radius where the radius is the maximum distance of a point to the center point c of cluster C, i.e. $r = \arg\max_{i\in C} d(x_i, c)$.

3.4 Tracking

In tracking, we need to ensure that the clusters identified from the previous scan are the same clusters in the current and all future scans, i.e., we require consistency of cluster assignment across all the scans. Using hierarchical clustering after the first scan did not yield a consistent assignment and was too time consuming for real-time tracking as it always started from scratch without utilizing data from the previous scan. We therefore had to think of a different clustering technique that makes it possible to utilize the hypotheses that we constructed

from the previous scan with a slight change that takes into account the motion model of the objects. The most natural choice would then be K-means which takes as input the number of clusters k and the initial center points. Instead of using the center points exactly as is from the previous scan, we use a simple linear prediction model that updates the centers to the expected new center based on the estimated velocity in the previous scans and the time between the scans, i.e. $x_{i+1} = x_i + \bar{v}t_\Delta$. This naturally speeds up the converge of the K-means algorithm.

There are two issues that we need to tackle separately in this stage. The first one is that an an object that is in motion might enter a blind spot and not be detected by the scanner, in which case we obtain an empty cluster and lose track of the object if we delete its hypothesis. Instead, we retain its hypothesis for a certain amount of time and update its center using the estimated velocity so that when it reappears, its hypothesis and cluster id are still present and are captured again. We have tested this method and it has worked for our dataset.

The second issue that we need to tackle separately is an emerging object that was not present there from the first scan and emerged in later scans, i.e. the number of clusters is k+1 rather than k. In order not to have this cluster merged into a cluster that represents a different obstacle, we compare the distance of any checked point in the assignment step to the radius, if its distance deviates from the closest cluster radius by a certain given threshold value then we don't assign it and keep it until after Kmeans has converged. For these unassigned points, we run hierarchical clustering again. The results of this twist are yet to be tested. In general Kmeans converged after 4-7 iterations and was an order magnitude faster than Hierarchical clustering.

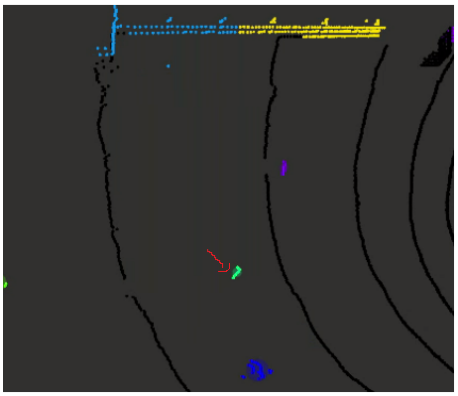
4 Conclusion

We now summarize what we've achieved in this lab. we have implemented a robust obstacle detection method that works for small objects as well as for big ones. We have aggregated the obstacle points into the individual objects that they belong to without any assumption about the order of transmission of the data points. We were also able to predict and track an object's next location in blind spots. We have tested our algorithm with a dataset that was captured by a UAV but with no objects in motion, as well as with a dataset that was captured by a stationary scanner but with objects in motion. Overall, the method can detect and track even small obstacles in a robust manner and should be able to deal with a UAV with objects in motion with some parameter tuning.

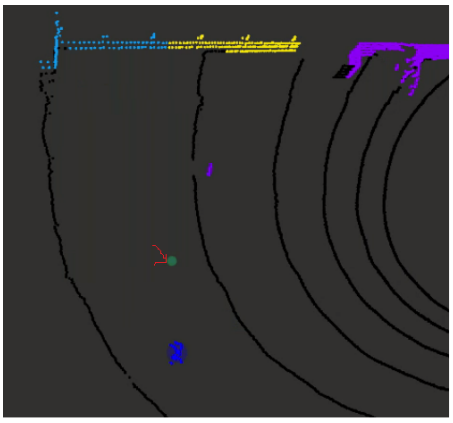
5 Future work

A future improvement would be incorporating mapping into our pipeline. This entails building smoothed 3D Maps from several scans and removing clutters from the map such that the algorithm is less susceptible to noise. We did not address the problem where the objects cross each other and could possibly merge

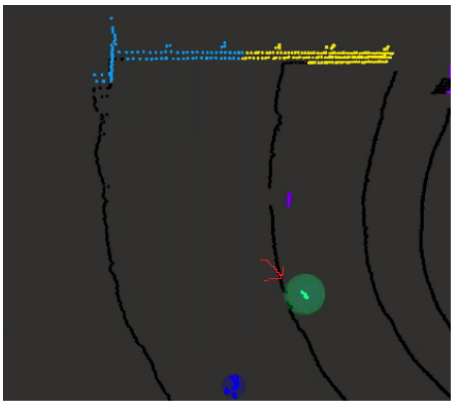
during clustering, but we believe that having incorporated a motion model into clustering should help with that. Basically by introducing the motion model we have applied a basic version of the Kalman Filter, and what we would like to do next is rely on the covariance matrix of the state estimate. We also would like to test our algorithm with data acquired by a UAV where both the UAV and the objects are in motion. Finally, we assessed the algorithm visually without quantifying the results. Quantitative results require having ground truth data such as labels that indicate for each pixel whether it's an obstacle or not.



(a)The initial cluster with a marker that represents its radius (marked by a red arrow)



(b)The object entered a blind spot but the hypothesis remained there



(b)The object after it exits the blind spot and reunites with its hypothesis again

Fig. 8. Outlining the algorithm tracking performance for different time steps.

Bibliography

Wolfram Burgard. Techniques for 3D Mapping . `http://www.acfr.usyd.edu.au/pdfs/projects/SLAM%20Summer%20School/Wolfram%20Burgard.pdf`, 2009. [Online].

Jacob Goldberger and Tamir Tassa. A hierarchical clustering algorithm based on the hungarian method. *Pattern Recognition Letters*, 29(11):1632–1638, 2008.

Michael Montemerlo, Jan Becker, Suhrid Bhat, Hendrik Dahlkamp, Dmitri Dolgov, Scott Ettinger, Dirk Haehnel, Tim Hilden, Gabe Hoffmann, Burkhard Huhnke, et al. Junior: The stanford entry in the urban challenge. *Journal of field Robotics*, 25(9):569–597, 2008.