RWU Hochschule Ravensburg-Weingarten University of Applied Sciences

Faculty for Electrical Engineering and Computer Science

PROJECT REPORT: TASK - 2

A SIMPLE OBJECT DETECTOR USING THE RADAR SENSOR

Guided by:

Dr. Professor Stefan Elser

*S*ubmitted by:

Prasad Sonani-36202

Master-Electrical Engineering and Embedded Systems

Akshay Chobe-36316

Master-Mechatronics

February 20, 2024

Contents

Chapter 1

Introduction

Object detection is a crucial task in many autonomous systems, such as self-driving cars, drones, and robotics. One of the key sensors used for object detection is the radar sensor, which can detect objects at long ranges and in challenging environments. The goal of this project is to develop a robust algorithm that can detect and track a moving object in a radar point cloud dataset. To achieve this, we propose a simple algorithm that can be implemented in four steps: isolating dynamic objects, finding moving point clusters, calculating the centre of the cluster, and evaluating the results.

1.1 Dataset and pre-processing

The dataset used in this project is a radar point cloud dataset collected by a pulsed radar sensor. The recording 1 consist 294 frames and recording 3 have 295 frames, each frame of radar sensor containing a 3D point cloud of the scene. Each point in the point cloud represents a radar measurement, and contains information such as the x, y, and z coordinates, radial velocity, and Intensity.

To check the efficiency of our intersection algorithm we will find centre for 240 frames of Radar sensor from recording 1 Dataset and compare it with the ground-truth from recording 1.

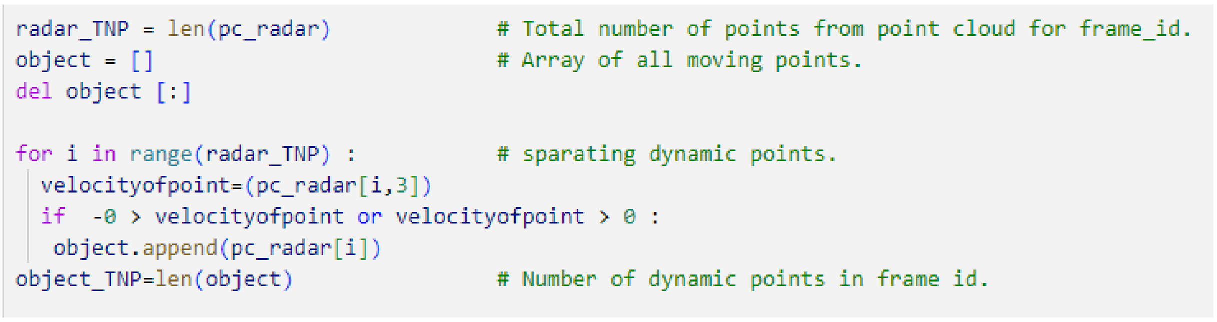
Overall, the project provides a detailed process for filtering and analyzing dynamic points in a radar point cloud dataset, which can be useful in various applications such as object detection and tracking in autonomous systems.

Chapter 2

Intersection Algorithm

2.1 Isolate Dynamic Objects

To separate the dynamic and static points from the radar point cloud , we implemented a velocity threshold of 0 m/s such that the points which don't have any velocity are considered as the static points and else are dynamic points. Those Dynamic points are stored in the array "object" as shown in figure 2.1



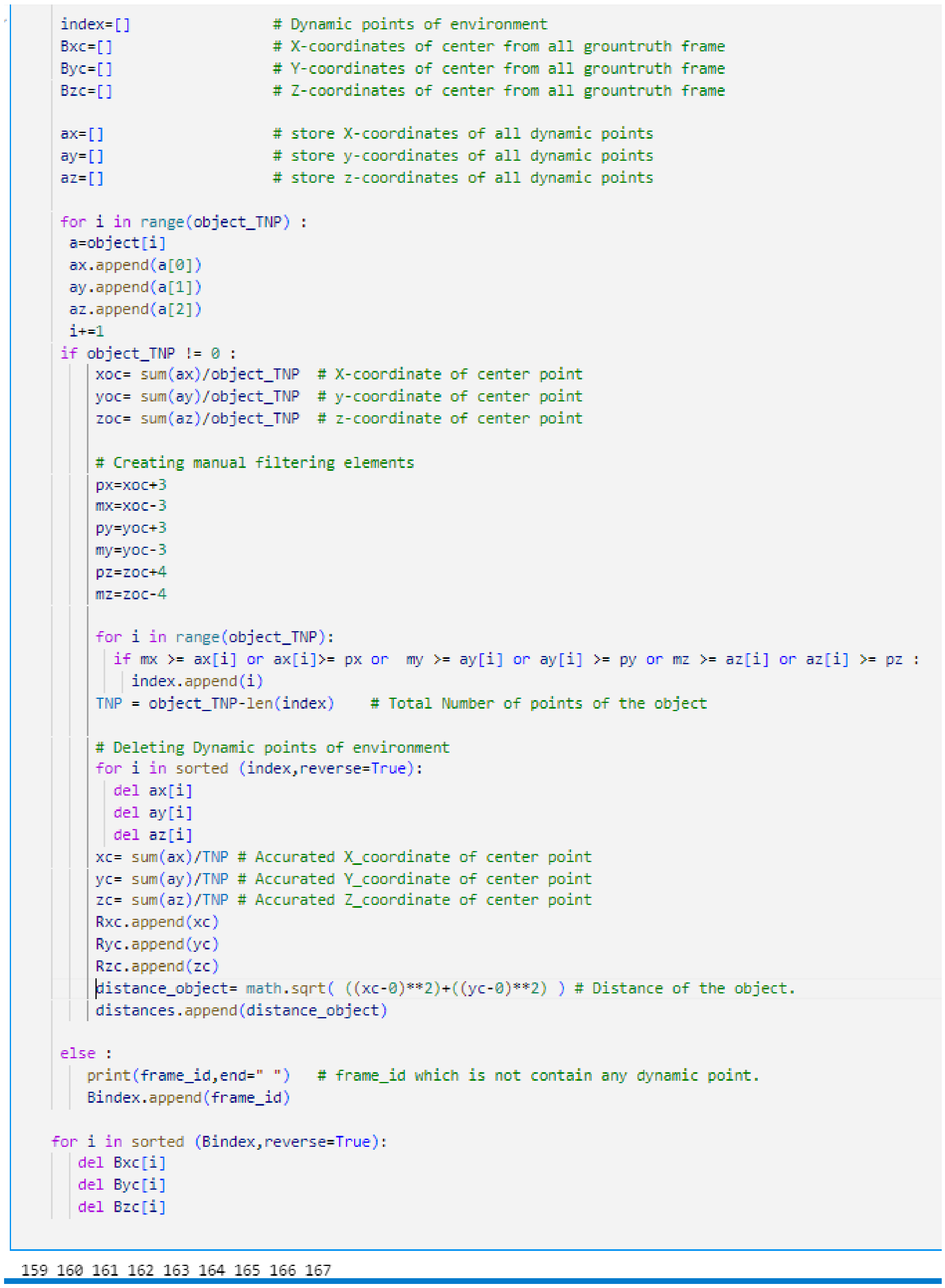
```
radar_TNP = len(pc_radar)             # Total number of points from point cloud for frame_id.
object = []                           # Array of all moving points.
del object [:]

for i in range(radar_TNP) :           # sparating dynamic points.
  velocityofpoint=(pc_radar[i,3])
  if  -0 > velocityofpoint or velocityofpoint > 0 :
   object.append(pc_radar[i])
object_TNP=len(object)                # Number of dynamic points in frame id.
```

Figure 2.1: Isolating Dynamic points

2.2 Filtering and calculating the centre of cluster

All the moving points in the array "object" contain the dynamic points from the surrounding environment and dynamic points from the moving object, so we imply filtering. Filtering is basically the process of separating out the irrelevant data points from the surrounding and focusing precisely on calculating the accurate centre of the object.



```
    index=[]                # Dynamic points of environment
    Bxc=[]                  # X-coordinates of center from all grountruth frame
    Byc=[]                  # Y-coordinates of center from all grountruth frame
    Bzc=[]                  # Z-coordinates of center from all grountruth frame

    ax=[]                   # store X-coordinates of all dynamic points
    ay=[]                   # store y-coordinates of all dynamic points
    az=[]                   # store z-coordinates of all dynamic points

    for i in range(object_TNP) :
     a=object[i]
     ax.append(a[0])
     ay.append(a[1])
     az.append(a[2])
     i+=1
    if object_TNP != 0 :
        xoc= sum(ax)/object_TNP  # X-coordinate of center point
        yoc= sum(ay)/object_TNP  # y-coordinate of center point
        zoc= sum(az)/object_TNP  # z-coordinate of center point

        # Creating manual filtering elements
        px=xoc+3
        mx=xoc-3
        py=yoc+3
        my=yoc-3
        pz=zoc+4
        mz=zoc-4

        for i in range(object_TNP):
          if mx >= ax[i] or ax[i]>= px or  my >= ay[i] or ay[i] >= py or mz >= az[i] or az[i] >= pz :
            index.append(i)
        TNP = object_TNP-len(index)    # Total Number of points of the object

        # Deleting Dynamic points of environment
        for i in sorted (index,reverse=True):
          del ax[i]
          del ay[i]
          del az[i]
        xc= sum(ax)/TNP # Accurated X_coordinate of center point
        yc= sum(ay)/TNP # Accurated Y_coordinate of center point
        zc= sum(az)/TNP # Accurated Z_coordinate of center point
        Rxc.append(xc)
        Ryc.append(yc)
        Rzc.append(zc)
        distance_object= math.sqrt( ((xc-0)**2)+((yc-0)**2) ) # Distance of the object.
        distances.append(distance_object)

    else :
       print(frame_id,end=" ")   # frame_id which is not contain any dynamic point.
       Bindex.append(frame_id)

   for i in sorted (Bindex,reverse=True):
      del Bxc[i]
      del Byc[i]
      del Bzc[i]
```

```
159 160 161 162 163 164 165 166 167
```

Figure 2.2: Algorithm for filtering and finding centre of cluster

To find the coordinates of the centre(xoc, yoc, zoc) of object We take the average of all the points in the array "object" to determine the coordinates of the object's center (x, y, and z), such that the average of all the points' x coordinates yields the object's x coordinate, and similarly for its y and z coordinates. The center of an object cannot be accurately determined because, as it is already known, irrelevant points from the surrounding environment are included in the array 'object' when calculating the center.

But eventually those centre point lies in or very close to the point cloud of the moving object because more number of dynamic point appears from moving object in radar detection.

In process of getting only object's points in point cloud, we will add manual boundaries in X,Y and Z coordinates and then we will find the center point. Here, those points are px,mx,py,my,pz and mz.

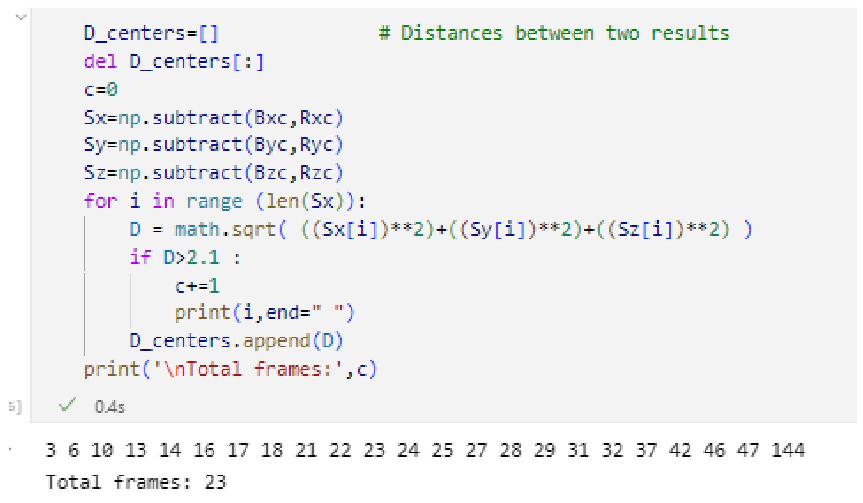
|px|+|mx| » width of object,
|py|+|my| » length of object,
|pz|+|mz| » height of object.
here, parameters of moving object are estimated values.

So, all points that are not inside the imaginary bounding box made by this six points are considered as the dynamic points from the environment. And those are stored in the Array "index" and then furthermore it will be removed by our algorithm. And we find the center point one more time with data of dynamic points of the object.

This center point (xc,yc,zc) is more accurate than the center point(xoc,yoc,zoc). Now, for calculating the distance of the object we only consider the accurate center point(xc,yc,zc). The array "distances" store distance of the object for all frame-id. The frames of radar sensor from recoding_1, who failed to detect dynamic points in point cloud are stored in array "Bindex". Those are frame-id 159 to 167. And those points will be removed by our algorithm.

2.3 Evaluate the result

In this case, we have a dataset that provides a ground truth for the position of the center point in each scene. This allows us to compare our results to the actual position and determine how accurate our method is. To find the accuracy of our intersection algorithm we will calculated the center points for all the frames of radar sensor from recording_1 and compare it with the center points given in the ground truth.



```
D_centers=[]              # Distances between two results
del D_centers[:]
c=0
Sx=np.subtract(Bxc,Rxc)
Sy=np.subtract(Byc,Ryc)
Sz=np.subtract(Bzc,Rzc)
for i in range (len(Sx)):
    D = math.sqrt( ((Sx[i])**2)+((Sy[i])**2)+((Sz[i])**2) )
    if D>2.1 :
        c+=1
        print(i,end=" ")
    D_centers.append(D)
print('\nTotal frames:',c)
```

```
3 6 10 13 14 16 17 18 21 22 23 24 25 27 28 29 31 32 37 42 46 47 144
Total frames: 23
```

Figure 2.3: Code for evaluating the result

Here Rxc, Ryc and Rzc are storing x,y,z coordinates accordingly of the centre points we calculated using our intersection algorithm for all frames of radar sensor from recording 1 as well as Bxc, Byc and Bzc stores centre points x, y and z coordinates accordingly from groundtruth of recording 1. Here we set 2.1 meter as our threshold value so all the frame ids which cross our threshold are printed in the result as shown in the figure 2.3

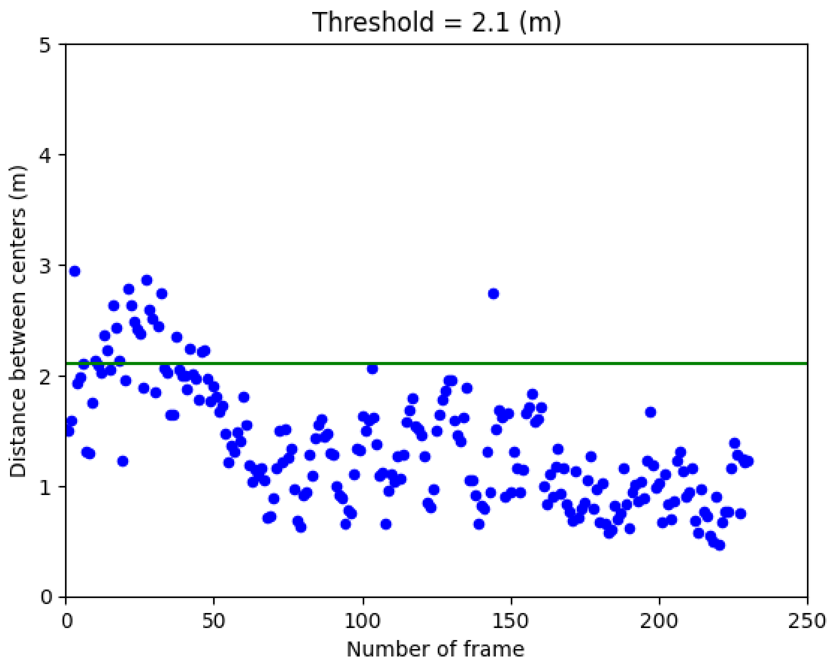


Figure 2.4: Implementation of threshold

Figure 2.4 demonstrates the point cloud of distance between centres with respect to all the frames after implementation of Filtering algorithm. As we have shown in the result of figure 2.3 and figure 2.4 that 23 frames are crossing threshold value of 2.1m among all

240 frames of recording 1.

So, accuracy of intersection algorithm $= \left[1 - \frac{23}{240}\right] \times 100 = 90.42\%$

2.4 Implementation of Intersection Algorithm

Upon determining the reliability of our algorithm on the labeled dataset, we applied it on the unlabeled data. To facilitate this, a new dataset, "Dataset_2022_unlabeled," is used. This dataset includes recordings of a car moving at a distance, in which the lidar sensor provides sparse data as previously observed. By utilizing this data, we can calculate or estimate the position of the car. Subsequently, by determining the car's position in each recording, we can calculate the distance of the car using intersection algorithm for radar sensor. This resulted in a visually informative plot, as the car in the new dataset moves at a low and consistent velocity towards the sensor.

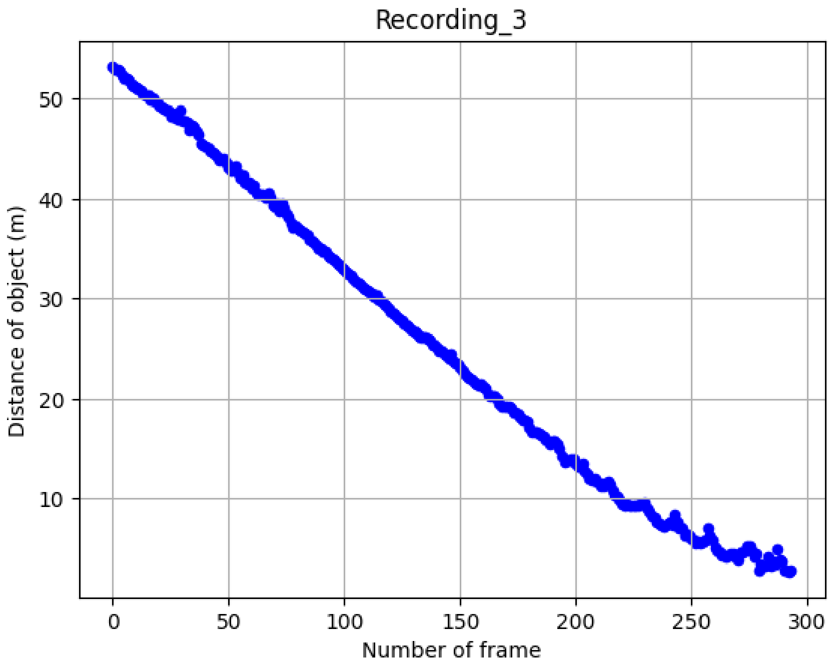


Figure 2.5: Intersection Algorithm on Recording_3

Here we can observe that the graph is smooth up to a distance of 10 meters away from the sensor. And as soon as the object comes closer to the sensor the graph starts to oscillate. The reason behind this is that the radar sensor is not able to capture accurate position of the object at the closer range. In other words we get distorted points of the object by radar sensor upto the close distance of 10 meters.

Chapter 3

Conclusion

In conclusion, this project has presented a simple object detector using radar sensor data. The algorithm is robust and can detect and track a moving object in a radar point cloud dataset.

The accuracy of our algorithm is more than 90% at threshold value 2.1m. The algorithm would need to be further optimized and tested in real-world scenarios before it could be used in a actual vehicle. However, the proposed algorithm serves as a strong foundation for further research and development purposes. We want to upgrade our intersection algorithm which will work with the dataset of all the sensors at a time, so detecting and measuring the distance of the object will become more accurate rather than using the dataset of only radar sensor. Multiple sensors should be used to overcome the shortcomings of individually using the lidar, radar and camera sensors.