DOCUMENT

SCORE

100 of 100

ISSUES FOUND IN THIS TEXT

0

PLAGIARISM

3%

Contextual Spelling	✓ No errors
Grammar	✓ No errors
Punctuation	✓ No errors
Sentence Structure	✓ No errors
Style	✓ No errors
Vocabulary enhancement	✓ No errors

DESIGN ORIENTED PROJECT

ROBOT NAVIGATION USING REINFORCEMENT LEARNING

SUBMITTED TO -
DR J.L. RAHEJA

SUBMITTED BY -
RIGVITA SHARMA
2016A7PS067P
SHRUTI SHARMA
2016ABPS0833P

Introduction

Abstract

Developing a proper navigation system requires an efficient object detection algorithm with tuning over a large dataset for training. However, this manual labelling of data could be sometimes expensive. Here we have worked on the acquisition of depth point cloud from a Kinect sensor and tried training a robot in a simulated environment (ROS-gazebo) using a dueling architecture based deep double-Q network (D3QN) for obstacle avoidance. The actual implementation and tuning on a simulated robot is performed using ROS and can be transferred to real robotic systems.

Problem Statement

Work on the conversion of RGB-D images into point cloud data.
Development of an object detection algorithm using ROS-PCL
Developing a dataset of depth PCD images using Kinect v2 for assisting in the implementation of a given algorithm on

an actual robot.
Development of an obstacle avoidance algorithm using reinforcement learning

Conversion into PCD
Dataset used -
PUTKK – PUT Kinect 1 & Kinect 2 data set, Mobile Robots Lab
Library used - Open 3D
Code snippet :

```
pcd = create_point_cloud_from_rgbd_image(rgbd_image,
PinholeCameraIntrinsic 1(PinholeCameraIntrinsicParameter
s.PrimeSenseDefault))
pcd.transform([[1, 0, 0, 0], [0, -1, 0, 0], [0, 0, -1, 0], [0, 0,
0, 1 2]])
write_point_cloud(str(file)+".pcd", pcd)
```

[1] Unoriginal text: 4 words
github.com/IntelVCL/Open3D/commit...

[2] Unoriginal text: 2 words
github.com/IntelVCL/Open3D/commit...

Object Detection using ROS-PCL
Dataset used -
Cornell RGB-D dataset with 24 labelled office scene point clouds.
The files are in .pcd format with fields - x, y, z, RGB, camera Index, distance_from_camera, segment_number and label_number. The label string corresponding to the label_number can be obtained from the file scene_labelling/scene_processing/labels.txt (for example, 1 is 'wall', 2 is 'floor', etc.).
Computational Tools used -
Robotic Operating Software - ROS
Point cloud library (PCL) is used to handle the 3-D point cloud data
Reading the data -
Every point cloud is stored using a pointer,
and the x,y,z coordinates are extracted.

Filtering
Downsampling :
It is used to reduce the number of data points.
A 3d voxel grid is created in the space, and all the points

present are approximated as the centroid of these points.
For, eg executing this on scene 1, we get :

Sparse outlier removal based on the computation of the distribution of point to neighbours distances in the input dataset.
We compute the mean distance from a point to all its neighbours, for all points. We assume that the resulted distribution is Gaussian with a mean and a standard deviation. Then, the mean and standard deviation can be considered as outliers and trimmed from the dataset for all points whose mean distances are outside an interval defined by the global distances.
Segmentation
Plane Model Segmentation
Does simple plane segmentation of a set of points,
that is found all the points within a point cloud that support a plane model
Used for the identification of planes such as ground.

Object Detection
Euclidean Clustering
After segmentation of planes such as walls and ground, we need to identify the objects or obstacles.
Space is divided into a 3D grid, using fixed width boxes, or more generally, an octree data structure - that is subdividing the space recursively into eight octants.
We find and segment the individual object point clusters on the plane.
Similar to the flood fill algorithm.

Results -

Kinect and Point cloud Dataset -
An indoor environment consisting of an indoor hostel scene was created with a sudden falling object
Kinect Studio was used to capture the coloured and 3-D point cloud video.
The video in .xef format was converted into frames and

stored as .mat files.
The recorded videos using Kinect v2 in the .xef format
The conversion into frames/ depth images -

Obstacle avoidance using deep reinforcement learning -
We use dueling network based deep double-Q network for obstacle avoidance.
What is D3QN?
Optimal Q-value function -
Here, we approximate the optimal Q-value function with deep neural networks, instead of computing the Q value function directly from a large state space.
Dueling Network - In traditional DQN, we estimate the Q-value of each action-state pair, given the current state using convolution layers. Then, only one stream of fully connected layers are constructed. However, in the dueling network, we have two streams of fully connected layers to compute the value and advantage functions separately. These streams are finally combined for computing Q-values. This two-stream dueling network structure is as -

Methodology -
The online network updates weights by back-propagation at every training step. However, unlike the online network, here the weights of the target network are set for a short period and then copied from the online network. Notably, the result state xt+1 is modified from both the online as well as the target network to calculate the optimal value Q 0∗ at time t +1. Then, with the discount factor γ and current reward rt, we receive the target value y at t. Finally, the error is calculated by deducting the target value with the optimal value Q ∗ predicted by the online network, given current state x, and is then back propagated to update the weights.
With the two techniques, the D3QN is expected to be more data efficient to speed up learning.

Model and training settings -
To train the network to produce a feasible control policy, robot actions are defined using linear and angular velocities in discretised form. r = v ∗ cos(ω)∗δt where v and ω are

local linear and angular velocity respectively, and δt is the time for each training loop, is defined as the instantaneous reward function. Here, δt is set to 0.2 seconds. The reward function is designed for the robot to run as fast as possible and be penalized by merely rotating on the same locality. The sum of the instantaneous rewards at all steps in an episode is the total episode reward. If there is a collision, we add an extra penalty of −10. Otherwise, the episode will continue until the maximum number of steps (Our experiments - 500 steps) and terminates with no penalties designed.

The environment -

Experimental results - in a simulated environment

Conclusion -

We have used D3QN based algorithm for obstacle avoidance using RGBD image as input. The training is done in a simulated environment. We have worked in a simulated environment, and the proposed algorithms can be easily applied to real-world robots to test the efficiency of the simulated model.

References

Towards Monocular Vision based Obstacle Avoidance through Deep Reinforcement Learning, Linhai Xie, Sen Wang, Andrew Markham and Niki Trigoni

Kinect and RGBD Images: Challenges and Applications, Leandro Cruz, Djalma Lucio, Luiz Velhoz, IMPA - VISGRAF Lab Rio de Janeiro, Brazil

https://code.tutsplus.com/tutorials/histogram-equalization-in-python--cms-30202

Open3D: A Modern Library for 3D Data Processing [3]

Qian-Yi Zhou Jaesik Park Vladlen Koltun Intel Labs

https://ythej.wordpress.com/2017/06/05/setting-up-kinect-v2-for-ubuntu/

https://github.com/georgeerol/RoboticPerception

http://incompleteideas.net/book/bookdraft2017nov5.pdf

http://wiki.ros.org/pcl/Tutorials

https://arxiv.org/pdf/1804.10332.pdf

Automatic Robot Navigation Using Reinforcement Learning, Amirhosein Shantia

[3] Unoriginal text: 9 words
github.com/IntelVCL/Open3D/pull/31...