Robotics Software Engineer Nanodegree: Slam Project

Manuel Huertas López

Abstract—In this project a 2D occupancy grid and a 3D octomap was created from a simulated environment provided by the Udacity team. In addition, a simulation environment was created from scratch, and the two previous maps were generated for this new environment. A robot provided with a Lidar, a RGB-D camera and odometry information from a differential drive was used, in the simulated environment. The library RTAB-Map was selected to solve the slam problem because: it offers a good performance and memory management, a good portfolio of development tools, and a good quality of the documentation. One of these tools, the rtabmap database viewer, was used to find out the closures and to identify the occupancy grid.

Index Terms—slam, udacity, rtab-map.

1 Introduction

THERE are some applications where a robot is provided with a map of its environment and it is able to estimates its pose based on this map, the odometry data and some sensor data like a Lidar or RGB-D camera. Furthermore, in some case the map is unknown, either because the area is unexplored or because the surrounding change often and the map may not be updated. In that case the robot must construct a map, the robot's pose is known and the robot is equipped with sensor data to measure its environment. Finally, in the most general case neither the Robot's pose and map is available, this is where slam cames in. In slam, the Robot will use the odometry and the sensor data to build a map of its environment while simultaneously localize itself relative to this map. The slam problem is quite challenging, with noise in the robot pose and measurement, the robot's pose will be uncertain and the construction of the map will be uncertaion as well, they are interrelated. This project will use an implementation of slam called real time appearance based mapping or RTABmap. This implementation is based on Graph Slam. GraphSlam is a slam algorithm that solves the full slam problem, this means that the algoritm recovers the entire path and map, instead of just the most recent pose and map.

2 Background / Formulation

2.1 Udacity

The Udacity provided data set, to be classified by the Neural Network, consists of a collection of objects of three types: candy boxes, bottles and nothing.

GoogleNet network was selected for several reasons [3]:

1) The image size fits well with one required by the network.
2) The network gives enough accuracy to pass requirements.
3) The number of operations, that is related to the inference time, is around 3 G-ops.

A classification network was selected with the following parameters:

- Training epochs[5]: how many passes throught the training data.
- Standard Network[GoogLeNet]: the predefined network achitecture.

2.2 Original Idea

The original idea selected was identified a coin in an image and to classify the coin, three different kinds of classes were used, corresponding to the European coins of 10 cents, 20 cents and one Euro.

GoogleNet network was selected for several reasons [3]:

1) The image size fits well with one required by the network.
2) The network gives a good accuracy with a reasonable inference time.
3) The network uses color image. For the Euros to be identify it is quite useful.
4) The number of operations, that is related to the inference time, is around 3 G-ops.

A classification network was selected with the following parameters:

- Training epochs[5]: how many passes throught the training data.
- Standard Network[GoogLeNet]: the predefined network achitecture.

3 Data Acquisition

The data set was collected using the camera of a mobile phone. The coins were put on a white paper and a video recording was started. Every few seconds the recording was paused to turn the coin randomly. Then, a tool to extract individual images from the video, ffmpeg, was used. The image were stored in jpeg format with a size of 1920x1080 pixels. A total of 2900 images, approximately, were collected.

In order to classify the coin a two-step strategy have been used. In the first step a classic feature extraction algorithm has been used. Due to the coins to be classified share a circular shape, the Hough Transform was used to find the coins in the image, classified them depending on the position in the image. The second step will use the inference, after training the network, to classify new images not seen before. The idea is to extract the region of the image corresponding to the coin and to inject this image as the input of the neural network.

It must be noticed that this is not a limitation for the real system, when trying to inference new samples, since the Hough Transform will be just used in this context to extract the portion of the image relative to the coin, and the neural network will inference the exact class for this new sample.

In addition, the principal advantage for this strategy is that the background of the image is not a problem during classification, because it is extracted, making this network quite robust to different environments.

Fig. 1: Example of raw image coming from the mobile camera.

A python script was created to take every image and, using the Hough Transform, to extract the coins from the images. The individual images were safe in directories with the name of the classes, in this case : coin10, coin20 and con100.

These images where down-sampled. The final image size, the ones used to train the network, was 256x256 pixels. The single images for every class was safe with the following directory structure:

```
data
├── coin10
│   ├── coin-10-image-1.jpg
│   └── coin-10-image-2.jpg
├── coin20
│   ├── coin-20-image-1.jpg
│   └── coin-20-image-2.jpg
└── coin100
    ├── coin-100-image-1.jpg
    └── coin-100-image-2.jpg
```

(a) 10 cents (b) 20 cents (c) 1 euro

Fig. 2: Example images coin extracted using Hough Transform.

4 Neural Network Trainning

Digits tools was used to train the network. The process of training the network in digits takes two steps. The following section show the configuration for both networks: the train the dataset provided by udaciy and the original idea.

4.1 Udacity

4.1.1 Create the Model

In the first step the model from the data set, classification type, was created. The following parameters were selected:

- Image type[Color]: color is a 3-channel RGB image.
- Image size(Width x Height) [256,256]: the image input will be resized to this value.
- Trainning Image: the folder with the structure as provided by udacity.
- %for validation[25%]: the percent of image to set apart for the validation process.
- %for testing[5%]: the percent of image to set apart for the test process.
- DatasetName[udacitydataset]: the name of the dataset for future references.

4.1.2 Train the Network

The network was trained with the model and network architecture previously defined using the digits platform.

Fig. 3: Trainnig chart Udacity.

4.2 Original Idea

4.2.1 Create the Model

In the first step the model from the data set, classification type, was created. The following parameters were selected:

- Image type[Color]: color is a 3-channel RGB image.
- Image size(Width x Height) [256,256]: the image input will be resize to this value. Since the original image was already of this size, resizing will not take place.
- Trainning Image: the folder with the structure describe in the previous section.
- %for validation[25%]: the percent of image to set apart for the validation process.
- %for testing[10%]: the percent of image to set apart for the test process.
- DatasetName[coindataset]: the name of the dataset for future references.

4.2.2 Train the Network

The network was trained with the model and network architecture previously defined using the digits platform.

Fig. 4: Trainnig chart original idea.

5 Results

5.1 Udacity

After training the network the result was evaluated, the result is on requirements.

Fig. 5: Evaluation of the model udacity.

5.2 Original Idea

After training the network a subset of the image reserved to test the result with the following output:

The results of the classification were really good.

Fig. 6: Confusion Matrix.

Fig. 7: Classification result of the network for testing images.

6 Discussion

It has been notice during the development of this project the importance of the data collection process. In order to have a good accuracy is quite important to have a big number of samples of every class. The process of acquiring these images can be a complex and time-consuming task. The background where the object were collected is also part of the acquisition problem. In this project a mix solution, using traditional image processing techniques, Hough Transform, was used to overcame this problem. The final accuracy obtained was quite good.

There are some parameters that must be taken in account depending on the final deployed hardware and the desire performance:accuracy, inference time, power consumption, memory use. [3]

There are relation between these parameters. For example,there are a hyperbolic relationship between the accuracy and inference time, in order to have a small increase in the accuracy the inference time will increase following a hyperbolic curve.

For that reason the network architecture must be selected taking in consideration the specific requirements for the concrete application. Furthermore, due to the possible limitations of the hardware where the solution will be deployed, and taking in consideration the frame per seconds to be inference and the accuracy desire, the correct network architecture must be selected.

In this project GoogleNet was selected due that the solution for both Udacity model and own model was on requirements. Furthermore, due to the possible limitations

7 Conclusion / Future work

The accuracy of the classification process achieved was really good. Due to time constraints the classes consists only in three coin and the same side. The final number of the classes is quite bigger than that. There are eight euros coins with two side each. One of the size is common for all the European countries but the other side is country specific. This makes the process of gathering the samples more difficult and time consuming. Once the network is trained with all the coin types and size a mobile application can be used to inference the coins in the day basis. The fact that the background is extracted will make this network quite robots again different environments: coins spread out over a pub's bar or over a counter in a book store.

References

[1] https://eu.udacity.com *Robotics Software Engineer Nanodegree program*.
[2] Adit Deshpande *The 9 Deep Learning Papers You Need To Know About*.
[3] Alfredo Canziani, Eugenio Culurciello,Adam Paszke *An analysis of deep neural network models for practical applications*.