HTBLuVA Wiener Neustadt
Höhere Lehranstalt für Informatik

DIPLOMARBEIT

Localisation via ML Methods

Ausgeführt im Schuljahr 2019/20 von:

Algorithm Ida Hönigmann	5AHIF
System Engineering Peter Kain	5AHIF

Betreuer / Betreuerin:

MMag. Dr. Michael Stifter

Wiener Neustadt, am September 25, 2019/20

Abgabevermerk: Übernommen von:

Eidestattliche Erklärung

Hiermit erkläre ich an Eides statt, dass ich die vorliegende Arbeit selbstständig und ohne fremde Hilfe verfasst und keine anderen als die im Literaturverzeichnis angegeben Quellen und Hilfsmittel verwendet habe. Insbesondere versichere ich, dass ich alle wörtlichen und sinngemäßen Übernahmen aus anderen Werken als solche kenntlich gemacht habe.

Wiener Neustadt am September 25, 2019/20

Verfasser / Verfasserinnen:

Ida HÖNIGMANN Peter KAIN

Contents

Acknowledgement

The authors would like to thank ...

Kurzfassung

asdf

Abstract

asdf

Chapter 1

Introduction

Author: Ida Hönigmann

Robots are getting more and more mobile. While a few years ago their usage was mostly limited to aid factory automation, robots have found widespread adoption in a multitude of industries, such as self driving cars and autonomous delivery drones. A challenge frequently encountered is navigating in unknown environments, which either requires the robot to sense specific characteristics of its surroundings or to communicate with some external system.
The problem of navigation has been looked at from many different angles. One popular approach in mobile robotics is to use the GPS, an external positioning system. In order to determine the position of a robot using the GPS, it has to establish communication with at least four satellites. The exact position of each satellite as well as the current time is broadcast by the satellites. By measuring the time needed for the signal to reach the robot, the position can be calculated up to three meters accurately.
However, in some cases positioning a robot using external positioning methods is not possible. In the case of the GPS this can be due to obstacles interfering with the radio signals send by the satellites, for example occurring inside a building. In comparison this work focuses on a system that can navigate in outdoor as well as in indoor environments.

1.1 Goal

The goal of this diploma thesis is to implement a system which can localize a robot using no other sensors than a camera. This limitation was purposely chosen as the system will be used by future robotic students at the HTBLuVA (Technical Secondary College) and many robot systems used in the field of education are only poorly equipped with sensors that are able to detect its environment. One sensor used in the field of educational robotics is the either already equipped, or easily mountable camera.
As part of this thesis the authors not only want to implement an easy to use API for future robotic students, but to also show the possibilities and advantages of machine learning in localisation.
In order to accomplish precise localisation in various different surroundings, the authors plan on implementing a neural network. The neural network should take images, taken by the

camera, as an input, and outputs the relative distance to any object shown in the images. By using machine learning the system should be less dependent on a specific situation or setup in comparison to different camera based localisation methods. For example the localisation should work on objects varying in size and shape, as well as in different situations of lighting.

1.2 Motivation

In July 2019 the two authors of this work participated in the aerial tournament at the Global Conference on Educational Robotics held in Norman, Oklahoma. One of the two main challenges encountered at this tournament was landing a drone next to some randomly placed object, which colour, shape and size was known in advance.
The second challenge the participants at the tournament faced was flying from one side of randomly placed cardboard boxes to the other. The cardboard boxes, representing a mountain, are placed in one of various configurations, one of which can be seen in figure 1.1.

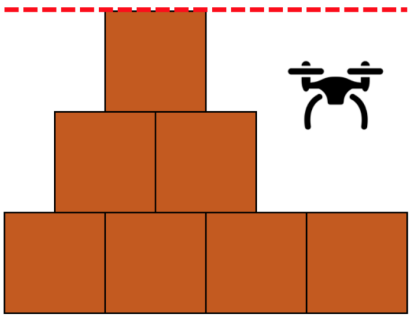

Figure 1.1: Seven cardboard boxes, representing a mountain, are placed in a random configuration. The team scores points if the drone passes the mountain while staying under the height limit indicated by the red dotted line.

The drones used at this aerial tournament are equipped with a camera, while lacking any other sensor that can be used to detect the obstacles and game items. Therefore the participants needed to be able to detect the distance to the object and the cardboard boxes using only the camera. At the Global Conference on Educational Robotics the authors of this work decided to detected the object based on its colour, but had to invest quite some time tweaking the values to get the localisation working correctly. Therefore the authors want to research and implement a method that is more robust than the colour based one.

1.3 Outlook / Perspective

The objective of this work is to create a system which uses machine learning methods in localising objects. After having trained the system, it should reliably return the x, y and z distances to an object, shown in two pictures taken from different angles.

It is planned that the distance will be measured from the second camera position to the centre of object, as seen in Figure 1.2.

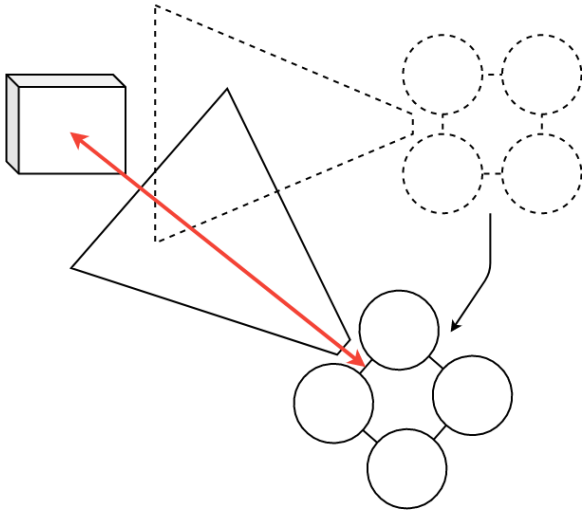

Figure 1.2: A drone tries to gather the data needed for the localisation of an object. After taking the first picture (dotted line) the drone flies to a second position (solid line) to take another picture from a different point of view. The red line indicates the distance vector to be returned.

1.4 Equipment

[TODO: Mit welcher Drohne soll das System funktionieren?, Kamera auf der Drohne]

Chapter 2

Study of Literature

Author:

2.1 Different Approaches to the Problem

[Input: Paper (gleiches Problem ohne NN) finden - Peter]
[Input: Video (eine Kamera, Entfernung zu Punkt (größe bekannt)) finden (ohne NN) - Peter]
[Vielleicht ist irgendetwas davon spannend: Links im Tex file]

2.1.1 LIDAR

2.1.2 Structure from Motion

2.1.3 Feature Tracking

2.2 Depth perception

[TODO: humans, two eyes - gleich wie bei unserem Aufbau]

2.2.1 Depth sensation

[TODO: Pigeons, deer, children (visual cliff)]

Chapter 3

Methodology

Author:

3.1 Stereo Camera

[warum verwenden wir keine stereo kamera?, weil keine vorhanden]

3.2 Image preprocessing

[schwarz-weiß? auflösung? filter? object detection?]

3.3 Neural Network

[TODO: Funktionsweise]

3.4 Generating data

[blender]

Chapter 4

ROS2

Author:

4.1 What is ROS?

[Ubuntu?]
[INFO: ROS (Robot Operating System) provides libraries and tools to help software developers create robot applications.[1]]
[TEST: asdf[2]] [Core Concepts]

4.1.1 Nodes

[Wie verwenden wir das? / Wie macht es unsere Arbeit leicher?]

4.1.2 ...

4.2 Why use ROS?

4.3 Comparing ROS and ROS2

[python2 < python3]

[1] Inc. Open Source Robotics Foundation. *Documentation - roswiki*. 2019. URL: `https://wiki.ros.org`.
[2] Open Source Robotics Foundation, *Documentation - roswiki*.

Chapter 5

Implementation

Author:

5.1 Generating test data

Good test data is of utmost importance in machine learning. The system can only know information that is depicted in the training data, which is why it is important to include as many aspects of the problem as possible in this data.
Since machine learning needs a lot of data in order to solve the given task it can be tiresome to generate and label all this data by hand. Therefore the authors decided to simulate the objects and the camera using a computer graphics modelling software called Blender.
Blender allows for relatively easy generation of training data by providing a Python API.
[TODO: Image camera setup, lightning, objects in Blender]
[TODO: Renders and labels for example objects]

5.2 OpenCV

OpenCV is a framework for image manipulation. Some of its use cases are changing the colour spectrum, filtering the image by colour and cropping images. The authors use OpenCV to test whether there are differences between filters for the images in the training data, for example greyscale images compared to coloured images.
[TODO: image manipulation und ...; aka besser beschreiben was OpenCV ist]
[TODO: wofür verwenden wir es? Ergebnisse zw. Graustufen und Farbbildern]
[TODO: Codebeispiele?]

5.3 Neural Network

The authors decided to use a software framework called TensorFlow for the first implementation of the neural network. This has the following two advantages: using Tensorflow allows for a low effort proof of concept and it makes testing out different configurations (e.g. number of hidden layers or filters in image preprocessing) of the neural network easier.

After it has been shown that the challenge of detecting the distance to an object can be solved using machine learning, the authors plan on implementing a neural network in C++ on their own. The knowledge gained in the TensorFlow implementation will be used in the C++ implementation, which hopefully will make the work less time consuming.

5.4 TensorFlow

As machine learning has gained popularity in recent years the demand for applicable frameworks grew. One of the most popular is called TensorFlow. It was developed by Google for internal use and was published under the Apache License 2.0 on the 9th of November 2015. TensorFlow supports APIs for Python, C, C++, Go, Java, JavaScript and Swift. Due to its popularity third party APIs for C#, R, Scala, Rust and many more were developed.
Its use cases reach from categorizing handwritten digits to YouTube video recommendations, one of the many applications Google use it for.
Tom Hope et al. describe TensorFlow as a software framework for numerical computations based on dataflow graphs[1].

5.4.1 Nodes and graphs

To compute a value using TensorFlow a computation graph has to be constructed. In this graph each node corresponds to an operation, such as subtraction or division. By connecting these nodes via edges the output of one node can be fed as input into another node. One example of such a computation graph can be seen in Figure 5.1.

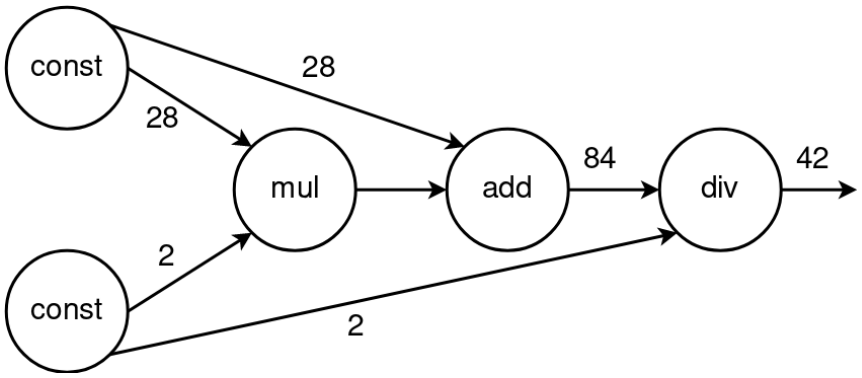

Figure 5.1: Each node represents an operation, where *const* stands for a constant value, *add* for addition, *mul* for multiplication and *div* for division. Edges, represented by arrows, connect nodes. The information shared between the nodes is described by the numbers written next to the edges. This computational graph calculates the result of the arithmetic expression $(28 * 2) + 28/2$.

[1]Itay Lieder Tom Hope Yehezkel S. Resheff. *Learning TensorFlow.* O'Reilly, 2017, page 6.

The implementation of the computation graph, shown in Figure 5.1, in Python could look as follows:

```
1  import tensorflow as tf
2
3  a = tf.constant(28)
4  b = tf.constant(2)
5
6  c = tf.multiply(a, b)
7  d = tf.add(a, c)
8  e = tf.divide(d, b)
9
10 with tf.Session() as sess:
11     out = sess.run(e)
12
13 print(out)
```

The first line specifies that the TensorFlow functionality should be imported. Line 3 and 4 define the two constant values and assigns them the values 28 and 2 respectively. In line 6 to 8 the other nodes of the graph are specified. E.g. in line 6 a new node, named c, is created and the output of node a and node b are connected as its input. To perform the calculation described by the graph a new session is created in line 10. Finally the output of the graph (node e) is specified in line 11, the result is calculated and printed in line 13.
TensorFlow allows for another way of specifying a graph with these arithmetic operations:

```
1  import tensorflow as tf
2
3  a = tf.constant(28)
4  b = tf.constant(2)
5
6  e = (a * b + a) / b
7
8  with tf.Session() as sess:
9      out = sess.run(e)
10
11 print(out)
```

This code is equivalent to the first one, but uses syntactic sugar to shorten line 6 to 8 in the first code block into line 6. At this point it should be noted that while it might look like it line 6 does not calculate anything. It simply describes how the computational graph should look. The answer (42) is calculated in the session in line 9.

5.4.2 Alternatives to Tensorflow

[INFO: Abstraction libraries such as Keras and TF-Slim offer simplified high-level access to the "LEGO bricks" in the lower-level library, helping to streamline the construction of the

dataflow graphs, training them, and running inference.[2]]

	Open Source	**Actively developed**	**Parallelization**	**Interface**
TensorFlow	Yes	Yes	Yes	Python, C, C++, Go, Java, JavaScript, Swift, R, Julia
Keras	Yes	Yes	Yes	Python
PyTorch	Yes	Yes	Yes	Python, C++
Torch	Yes	No	Yes	Lua, LuaJIT, C, C++, OpenGL
Wolfram Mathematica	No	Yes	Yes	Wolfram Language

[+ warum verwenden wir ausgerechnet Tenserflow]

5.4.3 Structure of our Neural Network

[wie lesen wir Daten ein, wie viele layer, was ist der output (maximale entfernung? z.B. 10m)]

5.5 C++ Implementation

5.6 Technical difficulties

[2] Tom Hope, *Learning TensorFlow*, page 7.

Chapter 6

Experiment 1

Author:

6.1 Environment

[TODO: viele Fotos] [TODO: Hintergrundfarbe, Untergrundfarbe, Struktur (Hintergrund und Untergrund), Beleuchtung (Art, Helligkeit, Richtung, mehrere Lichtquellen, welche Lichtquellen, ...)]

6.2 Setup

[TODO: viele Fotos] [TODO: welche Objekte (Größe, Farbe, wie viele (mindestens 3)?, ...), Kameras, Entfernung zu Objekten]

6.3 Sequence of Events

6.4 Results

Chapter 7

Lessons learned

Author:

Chapter 8

Experiment 2

Author:

8.1 Environment

[TODO: viele Fotos]

8.2 Setup

[TODO: viele Fotos]

8.3 Materials

[TODO: viele Fotos]

8.4 Sequence of Events

8.5 Results

Chapter 9

Conclusion

Author:

Index

Authors Index

Literature Index

Bibliography

Open Source Robotics Foundation, Inc. *Documentation - roswiki*. 2019. URL: `https://wiki.ros.org`.

Tom Hope Yehezkel S. Resheff, Itay Lieder. *Learning TensorFlow*. O'Reilly, 2017.