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 October 10, 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 October 10, 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.

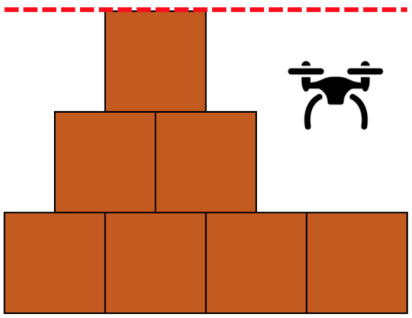


Figure 1.1: Seven cardboard boxes, representing a mountain, are placed in a random configuration. The team scores points if the drone passes the mountain while staying under the height limit indicated by the red dotted line.

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 detect 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.

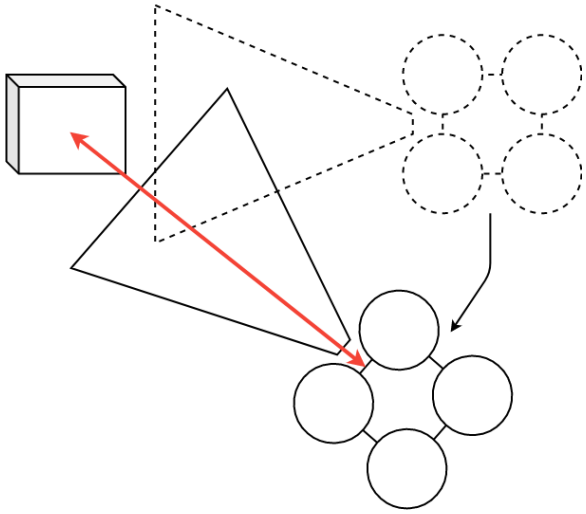


Figure 1.2: A drone tries to gather the data needed for the localisation of an object. After taking the first picture (dotted line) the drone flies to a second position (solid line) to take another picture from a different point of view. The red line indicates the distance vector to be returned.

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

The challenge of sensing distances to various objects has been solved using stereo vision cameras. Computer stereo vision systems use two horizontally displaced cameras to take two images which then are both processed together to gather the information on the depth of the images. This process can be rather complicated as the distortions (more specifically the barrel distortion and the tangential distortion) of the images have to be undone, before the two images are projected onto a common plane, a disparity map can be created by comparison of the two images and a 3d point cloud can be generated from it. In most robotics applications this point cloud is then filtered in search of some object, which distance was sought-after.

3.1.1 Distortion

Barrel distortion occurs when the lens used by the camera has a higher magnification at the centre of the image than at the sides. This distortion can be visualized as seen in Figure 3.1. To undo this distortion the pixel values in an undistorted image have to be calculated based on the pixel values in the distorted image.

$$r_u = r_d(1 + kr_d^2) \tag{3.1}$$

describes the calculation which computes the distance from the centre in the undistorted image (r_u) based on the distance from the centre in the distorted image (r_d) and some distortion parameter k, which is specific to the lens used. Gribbon et.al. note in their work[1] that this rarely is an integer value, therefore different equations are proposed:

$$x_d = x_u M(k, r_u^2) \qquad y_d = y_u M(k, r_u^2) \tag{3.2}$$

[1]Donald G. Bailey K.T. Gribbon C.T. Johnston. "A real-time FPGA implementation of a barrel distortion correction algorithm with bilinear interpolation". In: *Image and Vision Computing New Zealand*. 2003, pp. 408–413.

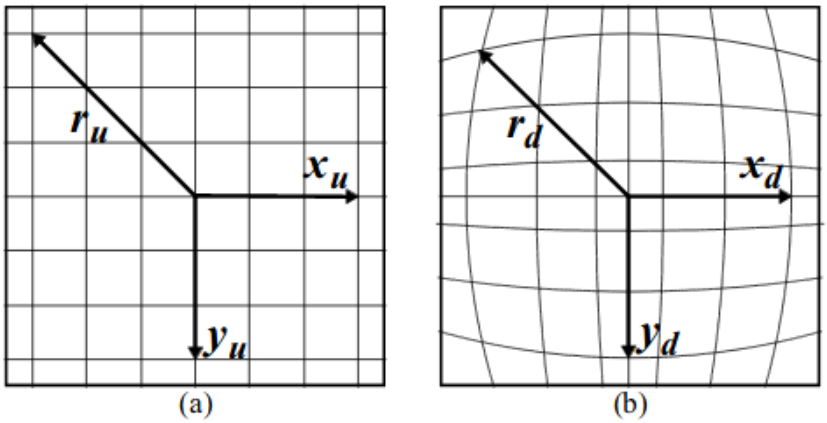


Figure 3.1: The left shows the original image composed of straight horizontal and vertical lines. On the right image the effect of the barrel distortion can be perceived, which causes the lines to curve toward the outside of the image, causing the lines to appear in a barrel like shape.[TODO: change image to own]

where the magnification factor $M(k, r_u^2)$ is

$$M(k, r_u^2) = \frac{1}{1 + k * M(k, r_u^2)^2 * r_u^2} \tag{3.3}$$

Tangential distortion in comparison to barrel distortion displaces points along the tangent of a circle placed at the centre of the image as seen in Figure 3.2.

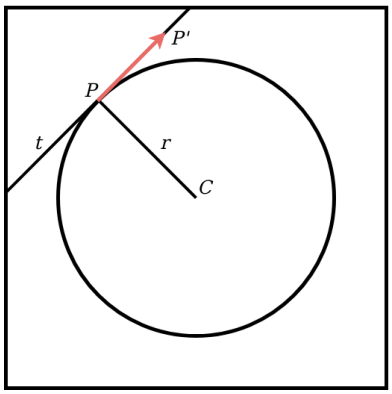


Figure 3.2: A point P is distorted along the tangent t of a circle placed at the middle of the image C with a radius r to a point P'. Distortions of this form are called tangential distortions.

The radius of the circle in Figure 3.2 is dependent on the point P. It can be calculated as the length between P and C. The length of the vector PP' is not uniform for all points and therefore depends on point P.

3.1.2 Image Rectification

Image Rectification projects multiple images taken from different points of view onto a common plane.

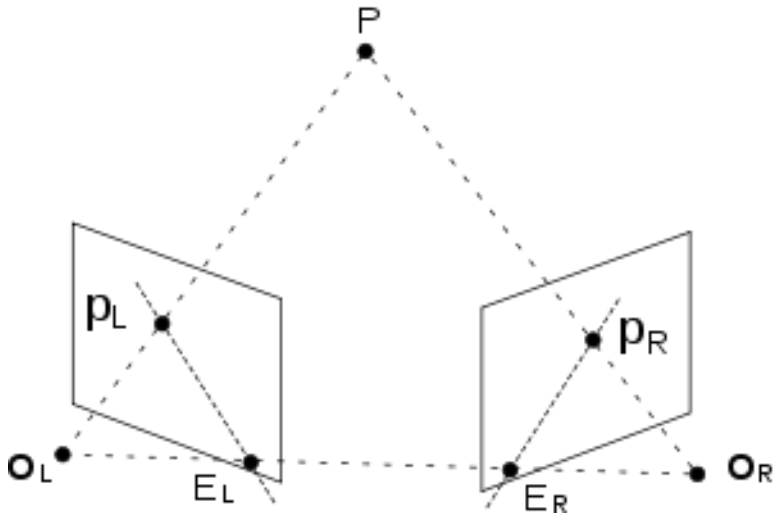


Figure 3.3: Two images containing some point P are taken from the two points O_L and O_R. Point P is projected in the image planes as points P_L and P_R. E_L and E_R depict the epipoles. [TODO: change image to own]

Chan et. al. propose an image rectification algorithm[2], which follows this sequence of events:

1. At least seven matching points visible on both images are found.
2. The fundamental matrix (as well as the epipoles) are estimated.
3. The common region is identified (using epipolar geometry constraints).
4. The epipolar line is transferred and the Bresenham algorithm[3] is used to extract pixel values.
5. The rectified image is resampled.

3.1.3 Disparity Map

3.1.4 3d point cloud

3.1.5 Challenges

Since many drones used in educational robotics can only carry a limited amount of weight it is not possible to attach a stereo camera to such a robot. Instead the drone has to take

[2]Hung Tat Tsui Zezhi Chen Chengke Wu. "A new image rectification algorithm". In: *Pattern Recognition Letters* 24.1-3 (2003), pp. 251–260.

[3]Jack Bresenham. "A linear algorithm for incremental digital display of circular arcs". In: *Communications of the ACM* 20.2 (1977), pp. 100–106.

the first image, fly to a second position located horizontally next to the first one and take the second image. This process in not as precise as a stereo camera, where the two lenses are always positioned at an exact interval from one another. Therefore the output of this system might not work as reliable. Additionally other factors, such as differences in the two images due to some time passing between the taking of the images can have an effect on the accuracy.
Therefore the authors try to approach this challenge from the machine learning point of view. Neural networks can be taught to work with different changes in the environment and still return results with a superior quality to conventional implementation.

3.2 Generating data

Neural Networks require huge amounts of data to work reliably. Because of the author's limited time frame this test data will be generated with the help of Blender 2.8. Blender is a free program for designing and animating 3D objects, which also supports scripting with python to add or remove objects from a scene. The authors will use this capability to generate the huge amounts of test data needed from the perspectives of the 2 cameras, which point to a specific object in the scene. This enables the authors to use and train a neural network, since shooting the amount of pictures needed by hand would take too long to consider this idea.

3.2.1 Blender

Besides 3D-modelling Blender enables the user to perform various different actions, such as laying out scenes, UV-Editing, shading, animating and rendering. Additionally scenes can be modified by executing Python scripts. Figure 3.4 shows the interface of Blender 2.8 with the default file loaded.
The authors decided to extensively use the scripting function in their work. One example of a Python script, that can be executed in blender is the following code:

```
1  import bpy
2
3  # Selects all cubes and deletes them
4  bpy.ops.object.select_all(action='DESELECT')
5  bpy.ops.object.select_by_type(type='MESH')
6  bpy.ops.object.delete()
7
8  # Adds a new cube
9  bpy.ops.mesh.primitive_cube_add(size=3, enter_editmode=False, location=(4, 2, 0))
10
11 # Adds a new material representing the colour red
12 bpy.ops.material.new()
13 material = bpy.data.materials[-1]
14 material.name = 'Red'
15 material.diffuse_color = (0.8, 0.1, 0.1, 1)
16
```

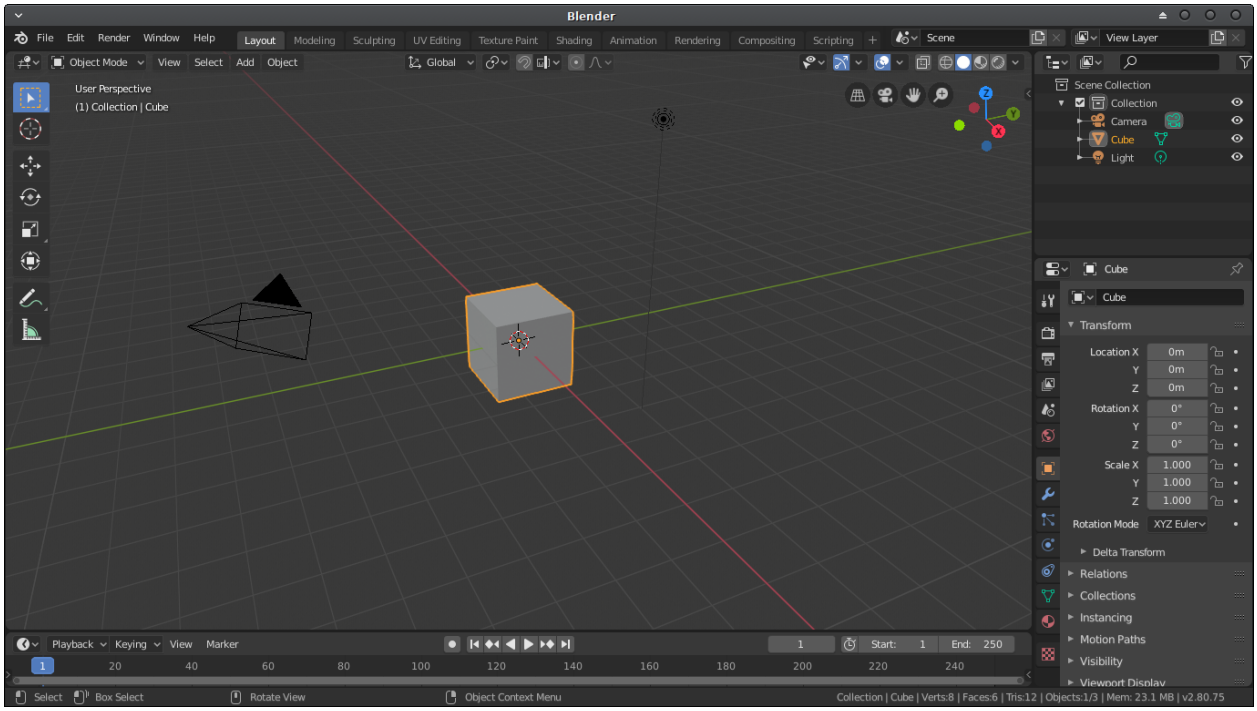


Figure 3.4: Blender interface after startup. The default scene features a grey cube, a camera (left of the cube) and a light source (black dot and circle positioned on the top right of the cube).

```
17 # Apply material onto the newly created cube object
18 bpy.context.active_object.data.materials.append(material)
```

This code first clears the scene from other meshes (because running the script twice would place the new cube inside the old cube). Then it adds a new mesh in form of a cube at the given location. Then we want to add some color to the cube. For this to work a material is needed, which is basically a specification how the surface of the object will look like. Advanced materials can represent raw or reflective surfaces, but in order to keep it simple this material will just represent a red surface (represented in red/green/blue/alpha channels ranging from 0.0 (for 0) to 1.0 (for 255)). Lastly the created material is applied to the object. The final result can be seen below:
[TODO: Add image]

3.3 Image preprocessing

After the test images have been rendered with the help of Blender, the authors decided to experiment if manipulating these test images would result in differences of distance perception by the neural network. This manipulation is done with OpenCV in Python3. OpenCV provides a lot of image manipulation tools; for this experiment the authors chose the following

methods of image manipulation:

3.3.1 Greyscale

A greyscale image is an image with only one value for the red, green and blue colour channels, resulting in different shades of grey instead of usual colours. This can easily be achieved by the following OpenCV code:

```
1 import cv2
2
3 image = cv2.imread('path/to/image')
4 image_greyscale = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
5 cv2.imwrite('path/for/saving/greyscale/image', image_greyscale)
```

This code first reads the image into 'image'. Then it converts the color of 'image' and stores the result into 'image_greyscale', which is then written to the specified path.

3.4 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.

3.5 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[4].

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

3.5.1 Computation graph

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 3.5.

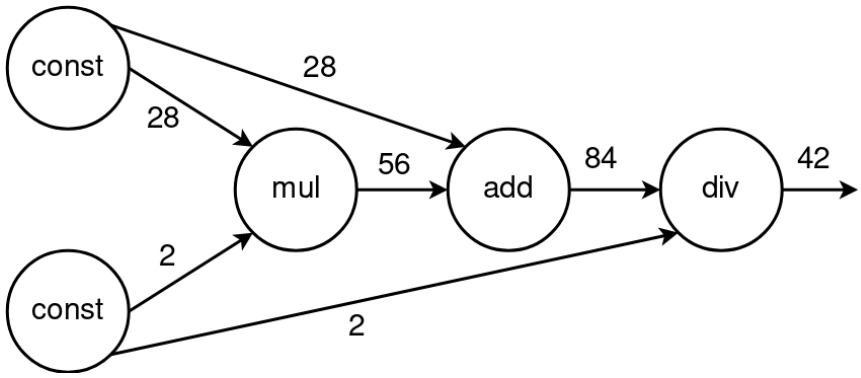


Figure 3.5: Each node represents an operation, where *const* stands for a constant value, *add* for addition, *mul* for multiplication and *div* for division. Edges, represented by arrows, connect nodes. The information shared between the nodes is described by the numbers written next to the edges. This computational graph calculates the result of the arithmetic expression $(28 * 2) + 28/2$.

The implementation of the computation graph, shown in Figure 3.5, 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.

3.5.2 Convonutional Neural Networks?

3.5.3 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.[5]]

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

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?]

5.2.1 Greyscale

[TODO: Input zweidimensionale Matrix statt dreidimensional (eine Zahl statt 3 pro Pixel)]

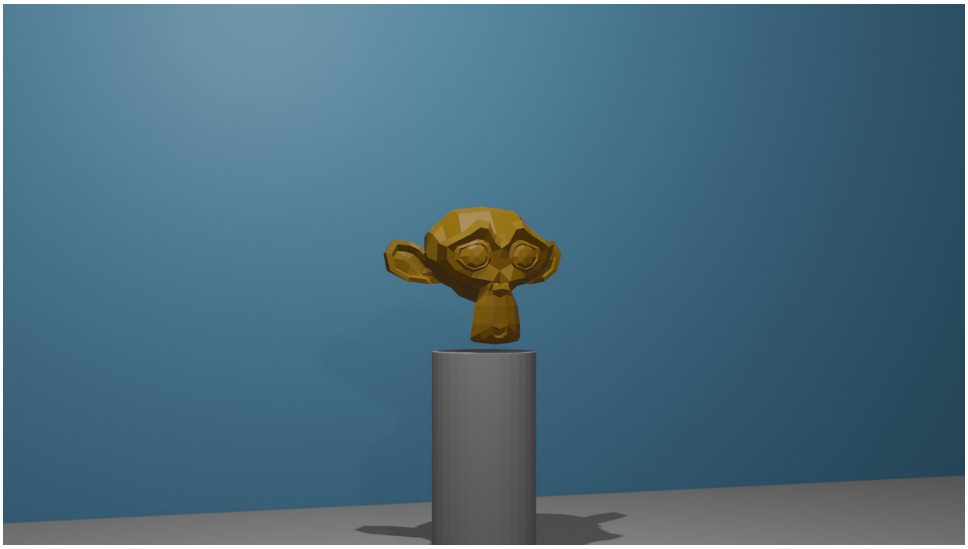


Figure 5.1: The original render.

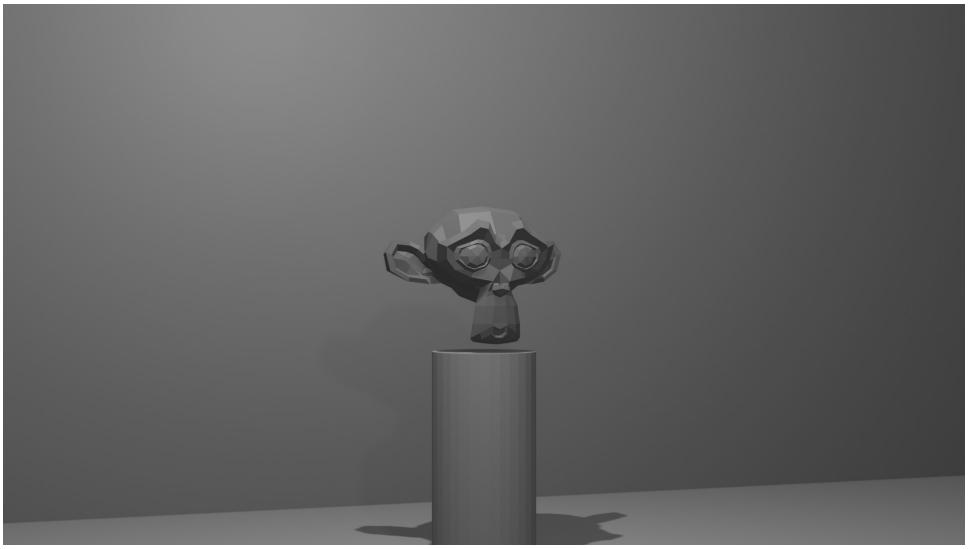


Figure 5.2: The greyscale image.

Figure 5.3: Comparison between normal and saturated image.

5.2.2 Saturated

5.2.3 Resolution

Figure 5.4: Comparison between normal and resolution image.

5.2.4 Brightness

Figure 5.5: Comparison between normal and brightness image.

5.3 Neural Network

5.3.1 Structure of our Neural Network

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

5.4 C++ Implementation

5.5 Technical difficulties

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

Bresenham, Jack. “A linear algorithm for incremental digital display of circular arcs”. In: *Communications of the ACM* 20.2 (1977), pp. 100–106.

K.T. Gribbon C.T. Johnston, Donald G. Bailey. “A real-time FPGA implementation of a barrel distortion correction algorithm with bilinear interpolation”. In: *Image and Vision Computing New Zealand*. 2003, pp. 408–413.

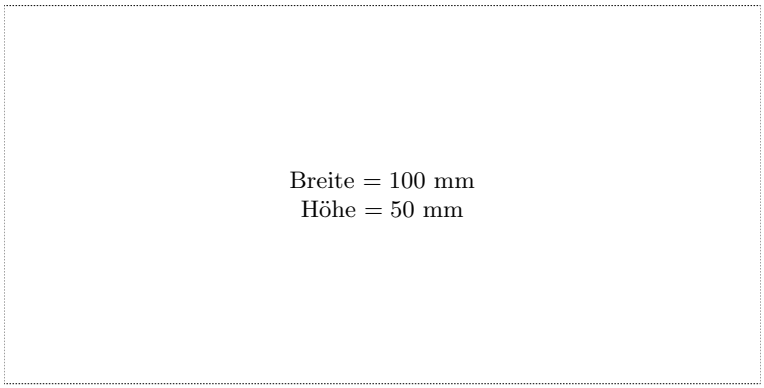
Open Source Robotics Foundation, Inc. *Documentation - roswiki*. 2019. URL: `https://wiki.ros.org`.

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

Zezhi Chen Chengke Wu, Hung Tat Tsui. “A new image rectification algorithm”. In: *Pattern Recognition Letters* 24.1-3 (2003), pp. 251–260.

Messbox zur Druckkontrolle

— Druckgröße kontrollieren! —



— Diese Seite nach dem Druck entfernen! —