

H T  
W I  
G N



**Hochschule Konstanz**  
Fakultät Informatik  
Institut für Optische Systeme

Eingereicht von  
Lukas Hornung  
Lukas Luschin  
Moritz Schmidt  
Timmo Waller-Ehrat

# Teamprojekt

## Mehrbildkamerasytem zur räumlichen Detektion von Modellhubschraubern

# Extended Abstract

Thema: Mehrbildkamerasystem zur räumlichen Detektion von Modellhubschraubern

Teammitglieder: Lukas Hornung, Lukas Luschin, Moritz Schmidt, Timmo Waller-Ehrat

Betreuer: Hochschule für Technik, Wirtschaft und Gestaltung HTWG Konstanz, Institut für Optische Systeme Prof. Dr. Georg Umlauf, Prof. Dr. Matthias O. Franz

Unser Projekt behandelt das räumliche Detektieren eines Modellhubschraubers. Die Detektion soll unter Laborbedingungen, das heißt, der Heli befindet sich vor einer weißen Wand, stattfinden. Mittels der Detektion soll auf einem Bild angezeigt werden, wo sich der Mittelpunkt des Helikopters befindet. Auch die Tiefe (Entfernung zur Kamera) des Helikopters soll ermittelt werden. Für diese Detektion sollen zwei oder mehr Kameras verwendet werden. Bei diesen handelt es sich um HIERKAMERAEINFÜGEN, die mit dem Computer über ein FireWire-Kabel verbunden sind.

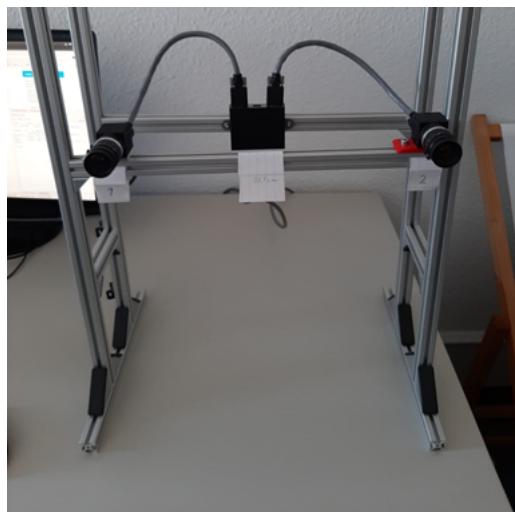
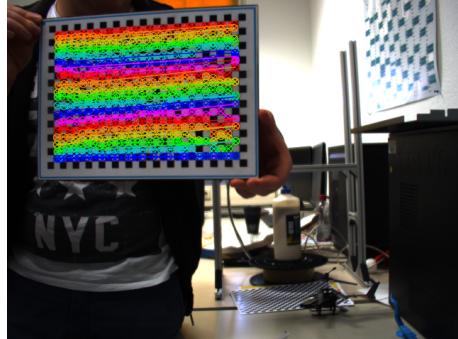
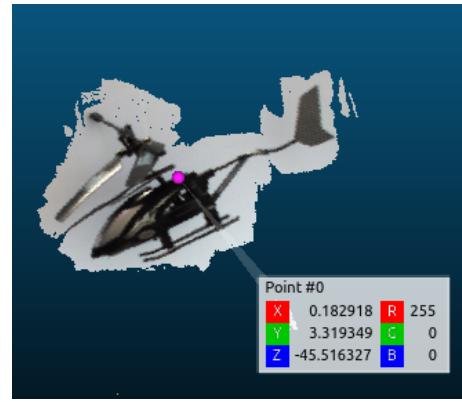


Abbildung 1: Kamera-System



(a) Kamera-Kalibrierung



(b) Helikopter-Punktwolke

Abbildung 2: Kamera-Kalibrierung und Punktwolke

Das Projekt wurde erfolgreich umgesetzt. Mittels zwei Kameras, die auf einer geraden Linie angebracht sind, kann der Mittelpunkt des Helikopters und dessen Abstand zur Kamera ermittelt werden.

Unser Programm kalibriert als erstes die Kameras einzeln und anschließend zu einander. Das Kalibrieren erfolgt über ein Schachbrett-Muster. Sind die Kameras zueinander kalibriert, kann mittels eines Feature-Detektors eine Punktwolke des Helikopters generiert und der Mittelpunkt berechnet werden.

Eine mögliche Erweiterung des Projekts wäre das Kalibrieren von zwei Stereo-Systemen zu einander, um eine noch höhere Genauigkeit zu erlangen. Dies wurde versucht umzusetzen, ist allerdings gescheitert.

# Abstract

Unser Projekt behandelt das räumliche Detektieren eines Modellhubschraubers. Die Detektion soll unter Laborbedingungen, das heißt, der Helikopter befindet sich vor einer weißen Wand, stattfinden. Mittels der Detektion soll auf einem Bild angezeigt werden, wo sich der Mittelpunkt des Helikopters befindet. Auch die Tiefe (Entfernung zur Kamera) des Helikopters soll ermittelt werden. Für diese Detektion sollen zwei oder mehr Kameras verwendet werden.

# Inhaltsverzeichnis

## 1 Einleitung

1.1	Aufgabenstellung und Zielsetzung . . . . .
1.2	Motivation . . . . .
1.3	Aufbau . . . . .

## 2 Technologien

2.1	Software . . . . .
2.1.1	OpenCV . . . . .
2.1.2	scikit-learn . . . . .
2.1.3	Open3D . . . . .
2.1.4	PyCapture . . . . .
2.2	Hardware . . . . .
2.2.1	Kamera . . . . .

## 3 Bildverarbeitung und Umsetzung

3.1	Kalibrierung . . . . .
3.2	Fehlerüberprüfung . . . . .
3.3	Tiefeninformationen . . . . .
3.4	Timmos Zeug . . . . .

## 4 Experimente

4.1	Vereinfachte Kalibrierung . . . . .
4.2	Tiefeninformationen . . . . .

## 5 Probleme

## 6 Fazit

# 1

## Einleitung

### 1.1. Aufgabenstellung und Zielsetzung

Im Rahmen dieses Teamprojekts stand die Entwicklung eines Mehrbildkamerasytems zur räumlichen Detektion eines Modellhubschraubers. Dies beinhaltet sowohl das Erkennen des Helikopters, als auch die Abstandsmessung von diesem.

Dies sollte mit Hilfe Bilderverarbeitungs- und Machine Learning-Techniken, sowie der Verwendung von zwei oder mehr Kameras umgesetzt werden.

Die Lernziele umfassten das Erlernen des Umgangs mit Kameras für die industrielle Bildverarbeitung, ein Verständnis für die Grundlagen industrieller Signalverarbeitung zu schaffen. Zudem sollten grundlegende KI-Verfahren erlernt werden.

### 1.2. Motivation

*„Computer vision, or the ability of artificially intelligent systems to see like humans, has been a subject of increasing interest and rigorous research for decades now.“*

— Naveen Joshi[4]

Das maschinelle Sehen gewinnt in den letzten Jahren immer mehr an Popularität. Sei es in der Forschung oder z.B. in der Spieleentwicklung mittels augmented reality.

### **1.3. Aufbau**

Durch die steigende Relevanz in der Praxis wurde auch unser Interesse für dieses Themengebiet geweckt. Es ist spannend zu verstehen, wie komplex die Dinge, die für uns Menschen selbstverständlich erscheinen, eigentlich sind. Ist es nur das ermitteln der Tiefe eines Objekts im Raum.

Ein weiterer Anreiz für das Projekt waren die verschiedenen angewandten Technologien. Wir alle interessieren und sehr für das Programmieren. Viel Erfahrung in der Programmiersprache Python hatte aber anfangs keines der Teammitglieder. Somit war das Erlernen dieser Sprache eine weitere Motivation.

Auch die zum Großteil verwendete Bibliothek OpenCV hat das Interesse an das Projekt geweckt.

### **1.3. Aufbau**

Die Ausarbeitung des Teamprojekts besteht aus drei Teilen.

Anfangs wird kurz auf die angewandten Technologien eingegangen. Anschließend wird die eigentliche Umsetzung und das Vorgehen erläutert. Zuletzt wird auf die aufgetretenen Probleme eingegangen und ein Fazit gezogen.

# 2

## Technologien

### 2.1. Software

#### 2.1.1. OpenCV

OpenCV ist eine Open-Source-Bibliothek, die über Algorithmen für maschinelles Sehen und Bildverarbeitung verfügt [10].



Abbildung 2.1: OpenCV

Quelle: [https://de.wikipedia.org/wiki/OpenCV#/media/Datei:OpenCV\\_Logo\\_with\\_text.png](https://de.wikipedia.org/wiki/OpenCV#/media/Datei:OpenCV_Logo_with_text.png)

#### 2.1.2. scikit-learn

Scikit-learn ist eine freie plattformunabhängige Python-Bibliothek, die für das maschinelle Lernen konzipiert ist. Die Software ist unter BSD lizenziert [11]. Von dieser Bibliothek wird lediglich die Implementierung des k-Means-Algorithmus verwendet.

## 2.2. *Hardware*



Abbildung 2.2: scikit

Quelle: [https://de.wikipedia.org/wiki/Scikit-learn#/media/File:Scikit\\_learn\\_logo\\_small.svg](https://de.wikipedia.org/wiki/Scikit-learn#/media/File:Scikit_learn_logo_small.svg)

### 2.1.3. Open3D

Open3D ist eine Open-Source Bibliothek, die diverse Algorithmen für das Verarbeiten von 3D-Daten bereitstellt.

### 2.1.4. PyCapture

Mittels PyCapture werden die Kameras angesteuert. Diese Bibliothek liefert 15 Bilder pro Sekunde.

## 2.2. *Hardware*

### 2.2.1. Kamera

# 3

## Bildverarbeitung und Umsetzung

### 3.1. Kalibrierung

Für eine Messung, bei der der Fehler minimiert werden soll, ist das Kalibrieren der Kameras unumgänglich. Durch die Linse einer Kamera entsteht eine tonnenförmige Verzeichnung. Diese Fehler sind meist so klein, dass sie vom menschlichen Auge nicht erfasst werden können [1] [3]. Durch die Kalibrierung der Kamera können diese kompensiert werden.

#### VERZEICHNUNGSEIGENSCHAFTEN

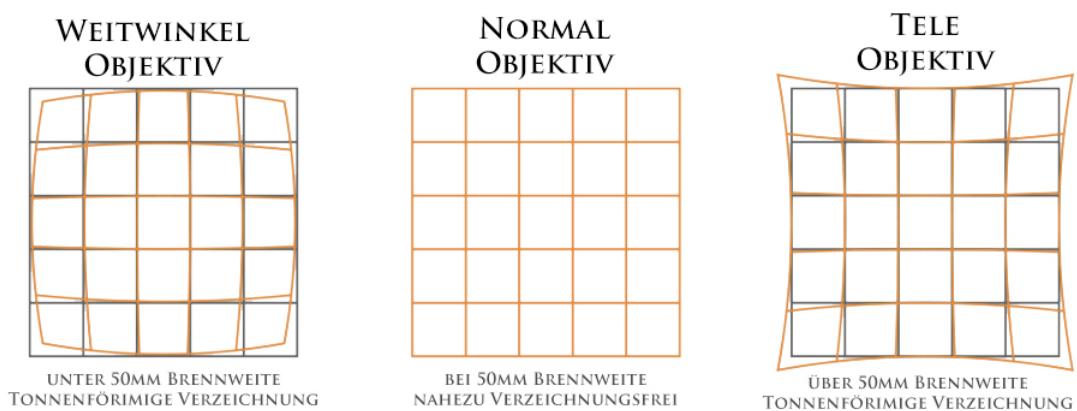


Abbildung 3.1: Verzeichnung

Quelle: <http://www.fotokurs-bremen.de/wp-content/uploads/2016/11/Objektiv-Verzeichnung.jpg>

### 3.1. Kalibrierung

Durch die Kamerakalibrierung werden folgende Parameter bestimmt:

**Intrinsische Parameter** Bezeichnen die Abbildung von 3D-Punkten im Kamerakoordinatensystem auf den 2D-Sensor der Kamera. Es sind Informationen der Kamera selbst, die unabhängig davon sind, wo sich die Kamera befindet und wie diese ausgerichtet ist [Intr].

**Extrinsische Parameter** Die räumliche Lage und Orientierung der Kamera zu einem Referenzkoordinatensystem, d.h. die Rotation und Translation [5] [9].

Da es sich bei dem System um ein Stereokamera-System handelt, ist die Kalibrierung von diesem etwas komplizierter.

Zuerst müssen die Kameras gesondert kalibriert werden. Dies wird mit der Funktion *calibrateCamera* von OpenCV durchgeführt. Für die Kalibrierung wird ein Schachbrett-Muster verwendet. Wichtig ist, dass bei der Kalibrierung beide Kameras dasselbe Bild verwenden. Für die Erkennung des Schachbretts wird die OpenCV-Funktion *findChessboardCorners* verwendet. Diese liefert die Objekt- und Bild-Punkte der Aufnahme. Bei den Objekt-Punkten handelt es sich um die 3D-Punkte des Bildes, bei den Bild-Punkten um die 2D-Punkte [6].

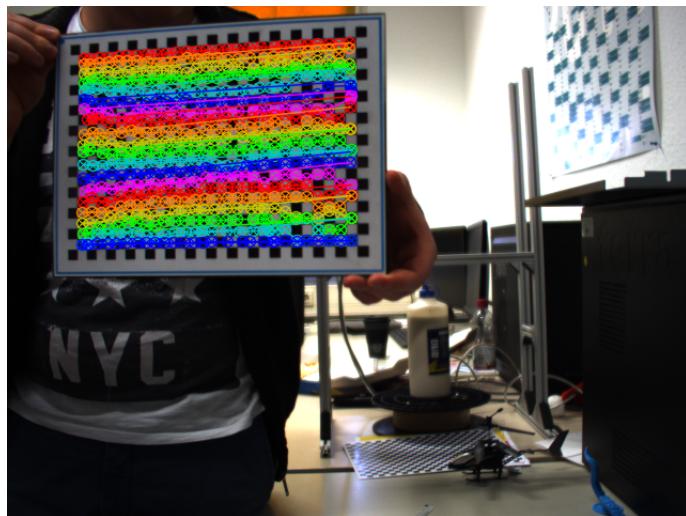


Abbildung 3.2: Kalibrierung

Für eine möglichst genaue Kalibrierung werden 50 Bilder verwendet. Anhand dieser wird jede Kamera mittels *calibrateCamera* kalibriert.

Die Funktion liefert die intrinsischen Parameter in Form von einer  $3 \times 3$  Kamera-Matrix

$$A = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

### 3.2. Fehlerüberprüfung

Wobei  $f_x$  und  $f_y$  die Brennweite in Pixeln und  $c_x$   $c_y$  ein Hauptpunkt, der normalerweise in der Bildmitte liegt, ist.

Die Ergebnisse dieses Vorgangs werden auf dem Computer gespeichert, sodass dieser nicht wiederholt werden muss. Anschließend wird das Ergebnis der Kalibrierungen an die OpenCV-Funktion `stereoCalibrate` übergeben.

Mit Hilfe der Stereo-Kalibrierung kann der Zusammenhang zwischen den Kameras ermittelt werden: Es werden von dem Bezugsbild, welches zum Ursprung des Koordinatensystems wird, die Objekt-Punkte verwendet. Von beiden Kamerasytem werden die Bild-Punkte, die jeweiligen Kameramatrizen und die Verzeichnungskoeffizienten verwenden.

Die für uns wichtigsten Ergebnisse der Stereo-Kalibrierung sind die Rotation und Translation der beiden Kameras zueinander.

## 3.2. Fehlerüberprüfung

Hier Reprojection-Error

## 3.3. Tiefeninformationen

Das Verwenden des Stereo-Kameras ist relevant für die Berechnung von Tiefeninformationen. Da die Information der Tiefe nicht in einem einzigen Bild ermittelt werden kann, wird eine zweite Kamera hinzugefügt. Diese ist im Raum verschoben, fotografiert aber zum größten Teil die gleiche Szene.

in 3.3, im rechten Bild, ist in grau der Ausschnitt zu sehen, der von beiden Kameras erfasst wird. Das Bild daneben zeigt den Versuchsaufbau: zwei Kameras, die horizontal verschoben sind. Somit erhalten wir den Normalfall, der wie folgt beschrieben wird: *Das achsparallele Stereosystem zeichnet sich durch zwei Kameras aus, die nur horizontal verschoben und deren Koordinatensysteme nicht gegeneinander verdreht sind [7].*

Nun ist es möglich, über die Ungleichheiten des überlappenden Bildbereichs Tiefeninformationen zu ermitteln. Diese wird mit Hilfe der Disparität berechnet.

Der horizontale Abstand, des gleichen Merkmals in beiden Bildern nennt man Disparität. *Die Disparität ist umgekehrt proportional zur Tiefe. [7].*

Durch die achsparallele Anordnung der Kameras, die nicht gegeneinander verdreht ist, kann die Z-Koordinate über die bekannten Kameraparameter, die Brennweite  $f$  der Kameras und der Basislänge  $B$  (Translation der Kameras zueinander), sowie die Disparität

### 3.3. Tiefeninformationen

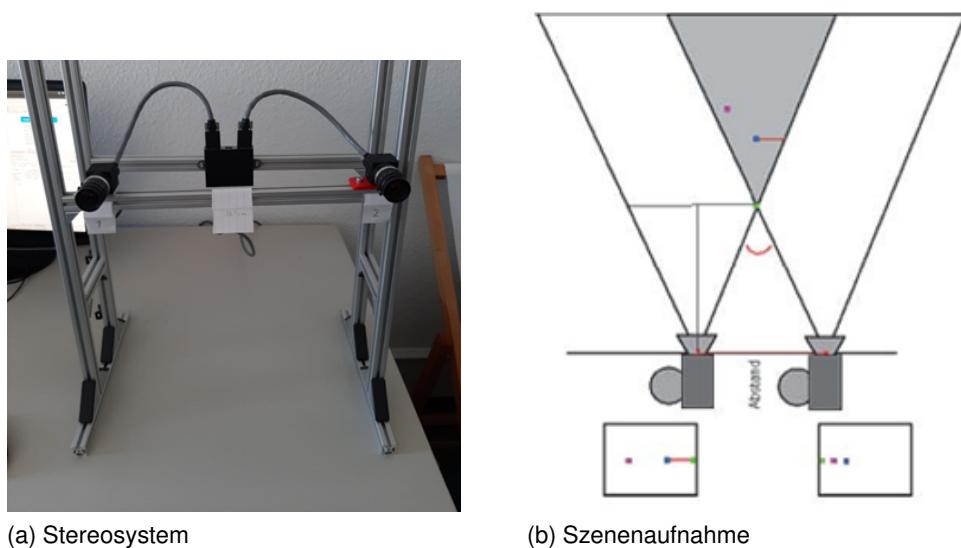


Abbildung 3.3: Stereo-System

Quelle: [https://me.efi.th-nuernberg.de/interaktion/index.php5/Bearbeitung\\_und\\_Gewinnung\\_von\\_Tiefeninformation\\_durch\\_Kopplung\\_zweier\\_Kameras](https://me.efi.th-nuernberg.de/interaktion/index.php5/Bearbeitung_und_Gewinnung_von_Tiefeninformation_durch_Kopplung_zweier_Kameras)

$D$  bestimmt werden.  $D$  wird wie in 3.4 schematisch dargestellt, durch die Bildpunktverschiebung der Beiden Punkte  $x$  un  $x'$  berechnet. Der Z-Achsenwert berechnet sich mit der Formel  $Z = \frac{f * B}{x - x'}$

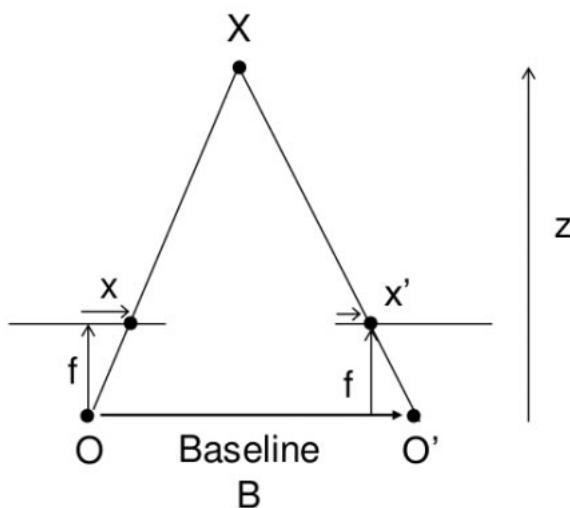


Abbildung 3.4: Tiefenberechnung

Quelle: [https://opencv-python-tutroals.readthedocs.io/en/latest/py\\_tutorials/py\\_calib3d/py\\_depthmap/py\\_depthmap.html](https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_calib3d/py_depthmap/py_depthmap.html)

3.4. *Timmos Zeug*

### **3.4. Timmos Zeug**

# 4

## Experimente

### 4.1. Vereinfachte Kalibrierung

Eine andere Möglichkeit, als in 3.1 erläutert, eine Kamera zu kalibrieren, ist mit Hilfe eines rechteckigen Kalibrierobjektes. In unserem Fall handelt es sich, wie in 4.1 zu sehen ist, um die Verpackung von Heftklammern.



Abbildung 4.1: Kalibrierobjekt

Die Formel zur Berechnung der Brennweite  $f_x$  und  $f_y$  ergibt sich aus den Formeln

$$f_x = \frac{\Delta x'}{x} z \text{ und } f_y = \frac{\Delta y'}{y} z$$

bei  $\Delta x$  und  $\Delta y$  handelt es sich um die jeweiligen Seitenlängen des Kalibrierobjekts.

$\Delta x'$  und  $\Delta y'$  sind die Längen der Seiten des Kalibrierobjekts in Pixelwerten.  $z$  ist der

## 4.2. Tiefeninformationen

Abstand zwischen Linse und Objekt [8] [2].



Abbildung 4.2: Vereinfachte Kalibrierung

Quelle: Solem 2012

Wie in 4.2 zu sehen ist, muss das Kalibrierobjekt senkrecht auf eine plane Oberfläche gestellt werden. So auch die Kamera. Die Kamera muss dann so ausgerichtet werden, dass das Objekt im Bild genau zentriert und entlang der Bildzeilen und -spalten ausgerichtet ist [2].

Obwohl diese Kalibrierung sehr ungenau erscheint, wurde ein dennoch relativ geringer Reprojection-Error, der den Wert  $\hat{?}$  hat, ermittelt. Der wie in [5] ermittelte Wert, von  $\hat{?}$ , ist dennoch besser.

## 4.2. Tiefeninformationen

Die Methode Tiefeninformationen zu ermitteln, wie es in 3.3 erläutert wird, wurde vorerst mit einem Experiment validiert.



Abbildung 4.3: Tiefe links

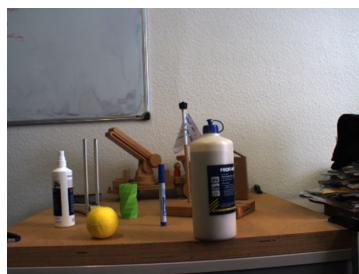


Abbildung 4.4: Tiefe rechts



Abbildung 4.5: Tiefe seitlich

Mit dem Stereo-System wurden zwei Aufnahmen einer Szene aufgenommen (4.3 und 4.4). Auf dem seitlichen Bild 4.5, sieht man die tatsächliche Tiefe der Objekte im Bild.

## 4.2. Tiefeninformationen

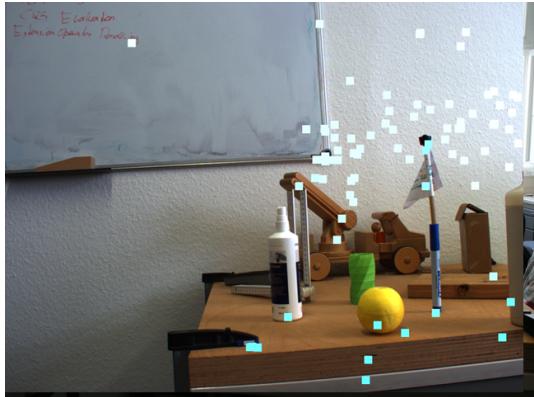


Abbildung 4.6: Tiefeninformation Punkte

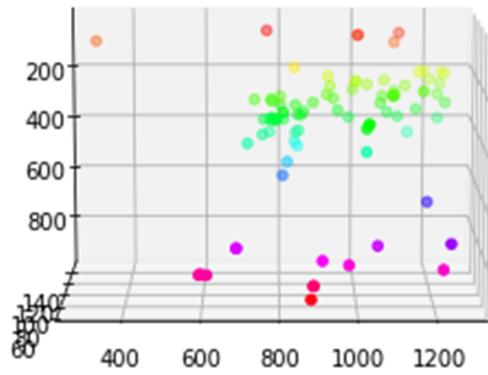


Abbildung 4.7: Tiefeninformation 3D-Plot

Auf den zwei Bildern der Szene wurden hier vorest händisch, dann mittels des SIFT-Algorithmus Referenzpunkte ermittelt, zu denen die Tiefe berechnet werden sollte. Der Abstand der beiden Kameras  $B$  wurde hier händisch mit einem Lineal ausgemessen. Der Wert hier für musste allerdings mehrere Male angepasst werden, da es wegen des Kamera-Gehäuses nicht klar war, wo genau sich der Sensor der Kamera befindet. Dann wurde die, wie in 3.3 erwähnte Formel angewandt, um den Wert der  $Z$ -Achse der jeweiligen Punkte zu ermitteln.

Mittels eines selbst geschriebenem Algorithmus, der Werte mit im Mittel großem  $Z$  weißlich und Werte mit im Mittel kleinem  $Z$  hellblau einfärbt, kann man sich die Informationen zur Tiefe auf einem Bild darstellen.

In 4.6 kann man gut erkennen, dass Objekte, die näher an der Kamera stehen, mit einem hellen blau markiert sind. Objekte, die weiter entfernt sind, mit einem weißlichem Farbton. Durch dieses Experiment war es uns möglich zu Validieren, dass unser Ansatz der Richtige war.

# 5

Probleme

# 6

Fazit

# Literatur

- [1] fotokurs bremen. *Objektive Brennweite und Verzeichnung*. URL: <http://www.fotokurs-bremen.de/objektive-brennweite-und-verzeichnung/> (besucht am 22.02.2020).
- [2] Matthias Franz. „ransac“. 2019.
- [3] Ingmar Jahr. *Kamerakalibrierung*. URL: <https://www.invision-news.de/allgemein/kamerakalibrierung/> (besucht am 23.02.2020).
- [4] Naveen Joshi. *The Present And Future Of Computer Vision*. URL: <https://www.forbes.com/sites/cognitiveworld/2019/06/26/the-present-and-future-of-computer-vision/#490553c0517d> (besucht am 22.02.2020).
- [5] Georg Rupert Müller. *Kalibrierung von Kameras*. URL: <https://www.unibw.de/tas/forschung/kalibrierung-von-kameras/view> (besucht am 23.02.2020).
- [6] OpenCV. *Calibration Tutorial*. URL: [https://docs.opencv.org/3.4/dc/dbb/tutorial\\_py\\_calibration.html](https://docs.opencv.org/3.4/dc/dbb/tutorial_py_calibration.html) (besucht am 22.02.2020).
- [7] Robert Sablatnig und Sebastian Zambanini. *Stereo and Motion*. URL: <https://www.cg.tuwien.ac.at/courses/EinfVisComp/Skriptum/SS13/EVC-18%20Stereo%20und%20Motion.pdf> (besucht am 24.02.2020).
- [8] Jan Erik Solem. *Programming Computer Vision with Python. Tools and algorithms for analyzing images*. 2012.
- [9] Springer. *Das Kameramodell*. URL: [https://link.springer.com/content/pdf/10.1007%2F3-540-27473-1\\_3.pdf](https://link.springer.com/content/pdf/10.1007%2F3-540-27473-1_3.pdf) (besucht am 23.02.2020).
- [10] Wikipedia. *OpenCV*. URL: <https://de.wikipedia.org/wiki/OpenCV> (besucht am 22.02.2020).
- [11] Wikipedia. *Scikit-learn*. URL: <https://de.wikipedia.org/wiki/Scikit-learn> (besucht am 22.02.2020).

# Abbildungsverzeichnis

1	Kamera-System	.	.	.
2	Kamera-Kalibrierung und Punktewolke	.	.	.
2.1	OpenCV	.	.	.
2.2	scikit	.	.	.
3.1	Verzeichnung	.	.	.
3.2	Kalibrierung	.	.	.
3.3	Stereo-System	.	.	.
3.4	Tiefenberechnung	.	.	.
4.1	Kalibrierobjekt	.	.	.
4.2	Vereinfachte Kalibrierung	.	.	.
4.3	Tiefe links	.	.	.
4.4	Tiefe rechts	.	.	.
4.5	Tiefe seitlich	.	.	.
4.6	Tiefeninformation Punkte	.	.	.
4.7	Tiefeninformation 3D-Plot	.	.	.