



Computer Vision Challenge

Disparity Map

TECHNISCHE UNIVERSITÄT MÜNCHEN
LEHRSTUHL FÜR DATENVERARBEITUNG

Dokumentation Gruppe 59

von

Vincent Mayer

Andreas Gaßner

Øivind Harket Bakke

Robert Lefringhausen

Theophil Spiegeler Castaneda

Juli 19, 2019

Inhaltsverzeichnis

1	Einleitung	1
2	Disparity Map	1
2.1	Epipolareometrie	1
2.2	Block Matching	4
3	Resultate	1
3.1	SAD	1
3.2	NCC	3
4	Unitests	1
5	GUI	1
6	Zusatzfunktionen	1
6.1	3D Rekonstruktion	1
7	Ausblick	2

Abbildungsverzeichnis

Literatur

1 Einleitung

Computer Vision ist in der heutigen Zeit ein sehr gefragtes Thema und besitzt sehr viele Anwendungsgebiete. Eines davon ist die Erstellung von Tiefenkarten welches die Aufgabe der diesjährigen Challenge im Kurs Computer Vision ist. Die Problemstellung ist aus zwei Bildaufnahmen der selben Szene aber von unterschiedlichen Perspektiven eine Tiefenkarte zu schätzen sowie die Rotationsmatrix und den Translationsvektor zuverlässig zu berechnen. Dafür sind uns die zwei Bilder und die calib.txt Datei gegeben. In der calib.txt Datei sind weiterhin wichtige Informationen über die Kamera gegeben welche benötigt werden um R und T zu berechnen sowie T auf Meter zu skalieren. Unser Programmablauf ist in Abb. 1.1 zu sehen.

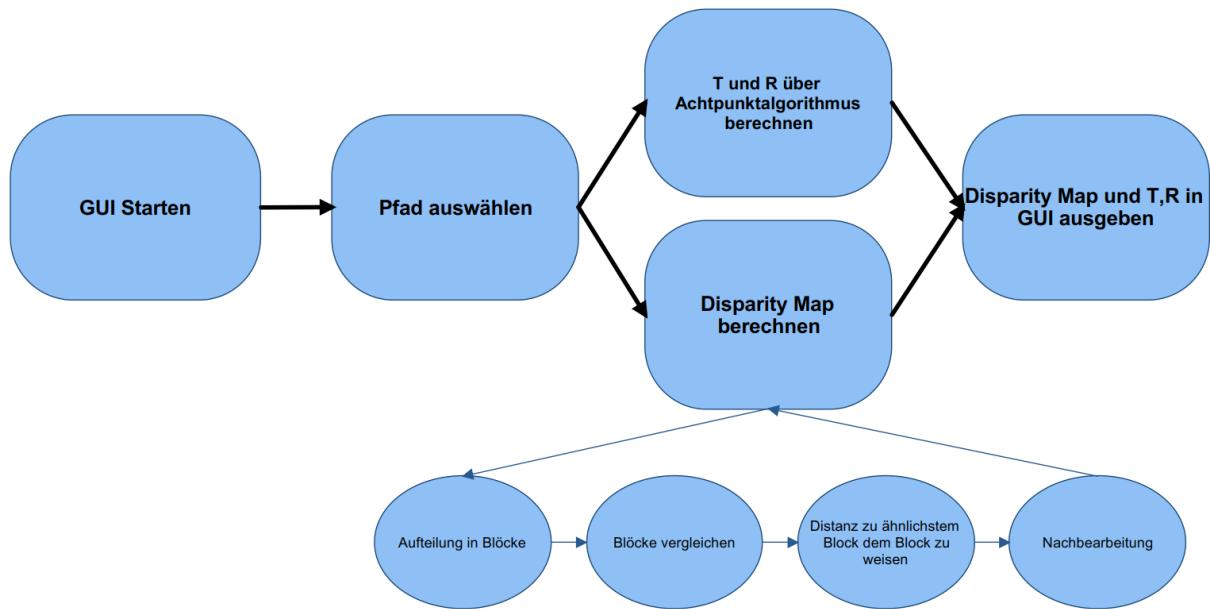


Fig. 1.1. Ablauf des Programms

2 Disparity Map

2.1 Epipolargeometrie

Eine Möglichkeit mit der das Tiefenverhältnis zwischen mehreren Punkten eines Bildes berechnet werden kann ist mittels Epipolaregeometrie. Dafür werden mehrere mathematische Schritte durchgeführt. Die einzelnen Schritte sollen in dem Flowchart in Abbildung 2.1 veranschaulicht werden. Diese wurden in der Vorlesung erläutert und werden in diesem Bericht als bekannt vorausgesetzt. Hier wollen wir auf unsere Überlegungen bezüglich der Eingabeparameter eingehen, die Algorithmen erläutern und die praktische Verwendbarkeit dieser Methode zur Erstellung einer Tiefenkarte bewerten.

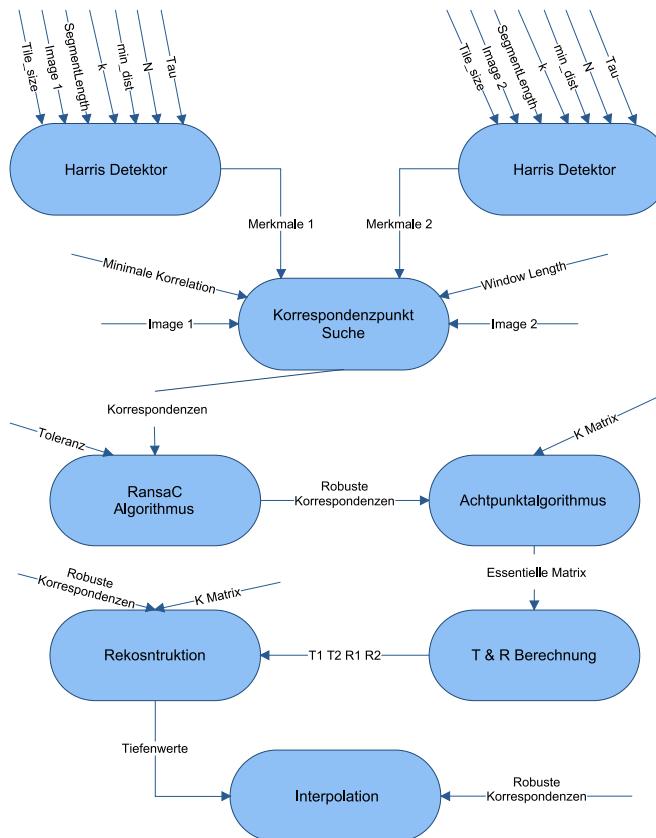


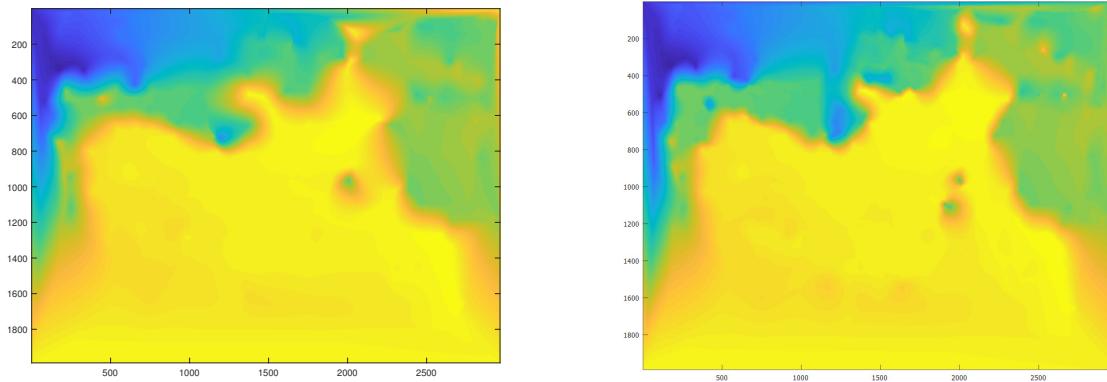
Fig. 2.1. Flowchart für die Berechnung von Tiefenwerten für Merkmalspunkte

Über den Harrisdetektor werden Ecken und Kanten erkannt und als Merkmalspunkte verwendet. Aus diesen Merkmalspunkten werden robuste Korrespondenzen zwischen den beiden Kameraansichten gesucht. Nur zu den robusten Korrespondenzpunkten kann das Tiefenverhältnis berechnet werden. Um möglichst viele Tiefenwerte (und damit eine gute Tiefenkarte) zu erhalten, sollte man schon am Anfang versuchen, möglichst viele Merkmalspunkte zu finden. Viele robuste Merkmalspunkte werden auch benötigt, um zuverlässig den T - Vektor und die R - Matrix zur euklidischen Bewegung schätzen zu können. Dies haben wir getan, indem wir den minimalen Abstand zwischen Merkmalspunkten auf den niedrigsten Wert gesetzt haben. Während die Segmentlänge ebenfalls auf den niedrigsten Wert ($3 \rightarrow 1$ über/unter Pixel) gesetzt wurde, um auch benachbarte Punkte zu betrachten. Die minimale Distanz und die Segmentlänge hängen somit voneinander ab. Es muss jedoch beachtet werden, dass nur die Merkmalspunkte aus welchen robuste Korrespondenzen gewonnen werden hilfreich sind. So kann eine große Anzahl an Merkmalspunkten nicht direkt mit mehr robusten Korrespondenzpunktpaaren in Verbindung gebracht werden. Als Beispiel haben wir den 'SIFT' (skaleninvariante Merkmalstransformation) Algorithmus getestet, welcher zwar viele Merkmalspunkte erkannt hatte, aus welchen jedoch keine robusten Korrespondenzen resultierten. Ein weiterer Wert, den wir benutzen können, um mehr Merkmalspunkte zu erhalten, ist Tau. Tau spiegelt den Schwellwert wieder, nach dem beurteilt werden soll, ob wir unsere Eigenwerte als ausreichend "groß" betrachten (bezogen auf Seite 14 von Kapitel 1.3 im Skript). Tau hat sich dabei als eine Variable mit der nicht viel an robusten Korrespondenzpunktpaaren gewonnen werden können erwiesen. Im Fall des Motorrads konnten wir mit einer Verringerung von $Tau = 10^6$ zu $Tau = 10^4$ lediglich 18 robuste Korrespondenzen dazugewinnen. Eine weitere Verringerung von Tau führt zu keiner Steigerung der robusten Korrespondenzen.

Letztendlich bleiben die zwei Parameter Tile Size und N übrig, welche ebenfalls dazu dienen sollen nicht zu nah aneinander liegende Merkmale zu finden. Damit können die robusten Korrespondenzen noch einmal deutlich erhöht werden, für die Tiefenkarte sind sie jedoch nur bedingt hilfreich, da die Information für einen nahe liegenden Punkt keinen großen Vorteil bringt. Die Verbesserung im Detail ist in Abbildung 2.2a und 2.2b dargestellt. Im Allgemeinen ist in Abbildung 2.2 eine Interpolation zwischen den Tiefeninformationen der robusten Korrespondenzpunktpaaren geplottet, plot 2.2a ist dabei aus 1425 robusten Korrespondenzen und plot 2.2b mit 4194 erstellt worden. Zu sehen ist eine Verbesserung im Detail wie z.B. zwischen Krümmer und Vordergabel. Die Eingabeparameter für die Korrespondenzsuche und den RansaC Algorithmus groß zu verändern ergibt kaum Sinn da eine Veränderung dieser Werte das Ergebnis verfälschen könnten. Nachdem die Lambdas aus der Rekonstruktion berechnet wurden, weisen wir den robu-

sten Korrespondenzen ihre Tiefenwerte zu und interpolieren zwischen diesen Punkten. Um ein Bild in Größe des Originalbildes zu erhalten, suchen wir die am nächsten liegenden Korrespondenzpunkte zu den vier Kanten und weisen der Kante den korrespondierenden Tiefenwert zu.

Es hat sich gezeigt, dass sich diese Variante nur für große Bilder in denen viele Merkmalspunkte gefunden werden, als praktisch anwendbar erweist. Diese Feststellung wird an den Abbildungen 2.3 - 2.6 deutlich. Deshalb haben wir uns für eine andere Herangehensweise entschieden. Diese errechnet die Tiefenkarte über den so genannten Block-Matching-Algorithmus, welcher im folgenden Kapitel erläutert wird.



(a) Tiefenkarte des Motorrads mit $N = 50$ & $til_size = [200 \ 200]$ (b) Tiefenkarte des Motorrads mit $N = 300$ & $til_size = [100 \ 100]$

Fig. 2.2. Tiefenkarte des Motorrads mit (a) 1425 und (b) 4194 robusten Korrespondenzen

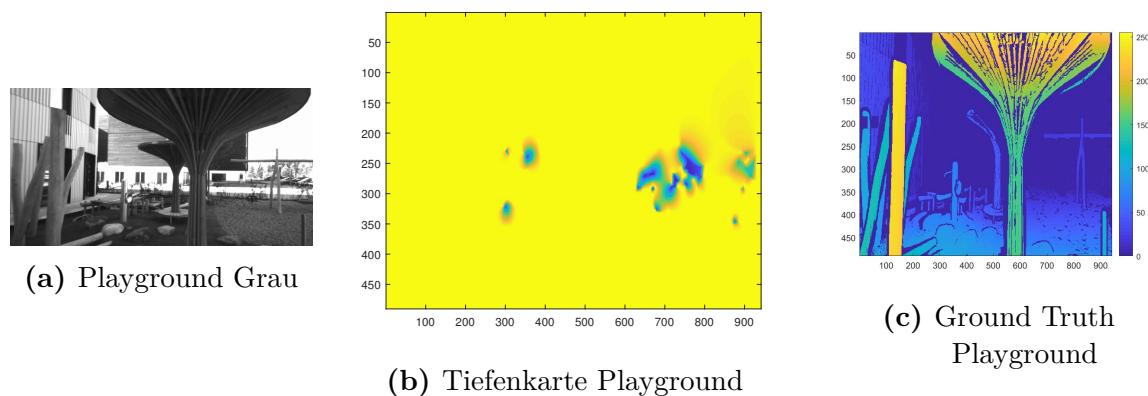


Fig. 2.3. Playground 643 robuste Korrespondenzen

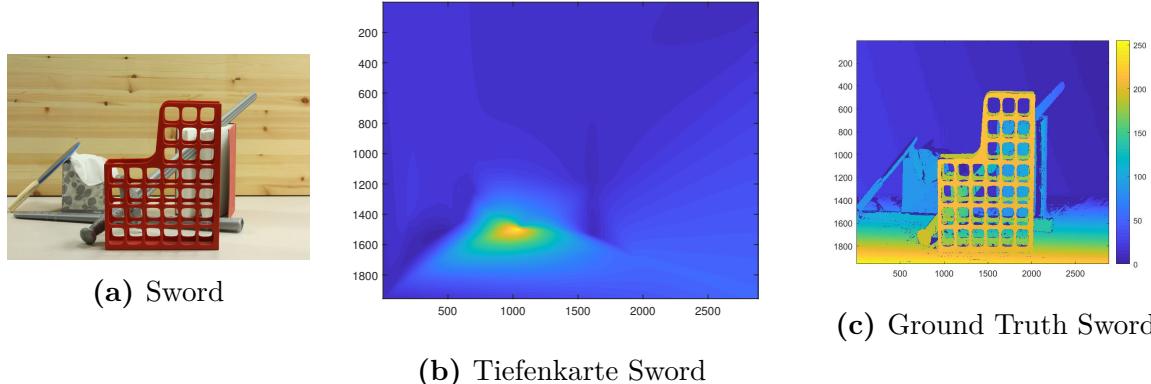


Fig. 2.4. Sword 46 robuste Korrespondenzen

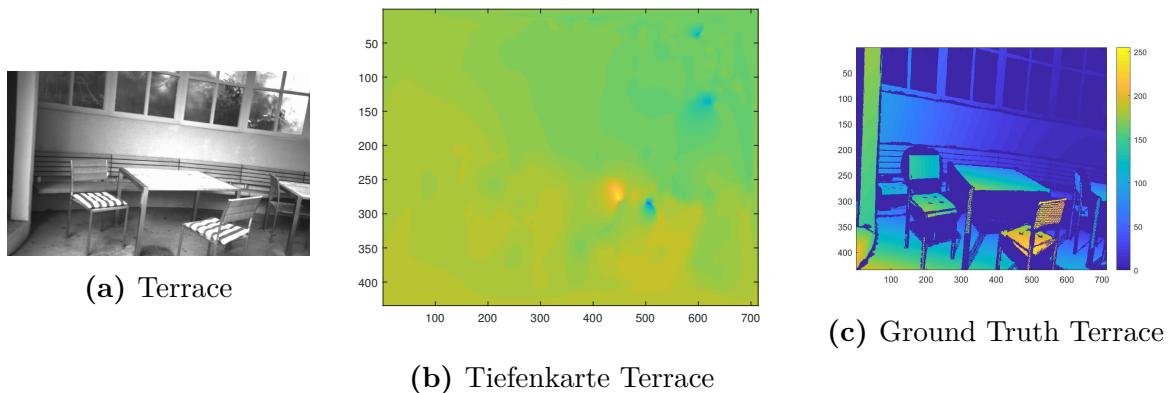


Fig. 2.5. Terrace 617 robuste Korrespondenzen

2.2 Block Matching

Block Matching hat zum Ziel für jeden Bildpunkt aus einer Stereoaufnahme den korrespondierenden Punkt im anderen Kamerabild zu erkennen. Dies ermöglicht es generell eine detaillierte Tiefenkarte für beide Kameraaufnahmen zu generieren. Die Suche nach korrespondierenden Bildpunkten wird mit Hilfe der sogenannten epipolaren Einschränkung begrenzt. Diese besagt, dass der korrespondierende Bildpunkt in der zweiten Aufnahme x_R zu einem Bildpunkt der ersten Aufnahme x_L auf der Epipolarlinie e_r liegt. Der Sachverhalt ist in Abb. verdeutlicht Dadurch muss nicht für jeder Bildpunkt der einen Aufnahme mit jedem Bildpunkt der anderen Aufnahme verglichen werden. Alle zur Verfügung stehenden Bilder sind zudem rektifiziert, was bedeutet, dass sämtliche Epipolarlinien horizontal durch das Bild laufen. Der Suchraum für den Korrespondenzpunkt in der zweiten Aufnahme zum Bildpunkt der ersten Aufnahme kann entsprechend entlang

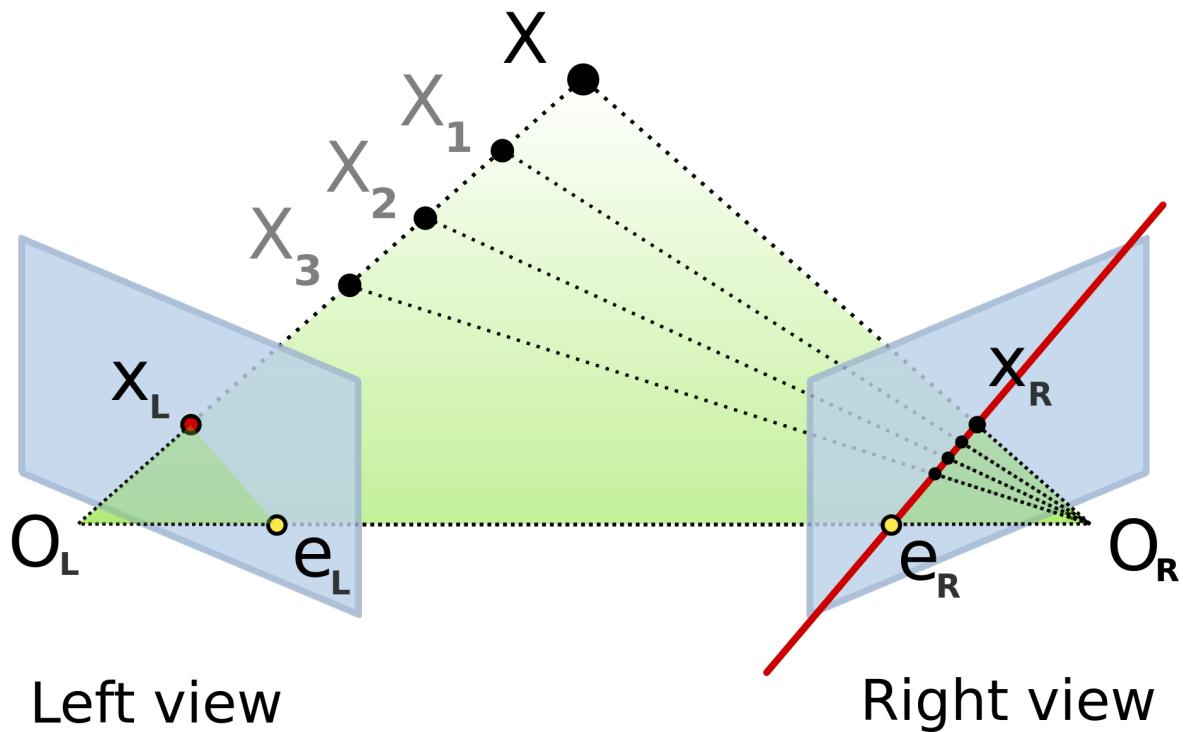


Fig. 2.6. Epipoar Einschränkung [1]

einer horizontalen Linie stattfinden. Diese horizontale Linie besitzt die selbe $y-$ Koordinate, wie der Bildpunkt in der ersten Aufnahme. Als Grundlage zur Implementierung des Block-Matching-Algorithmus dient Referenz [2]. Der Algorithmus sucht, wie bereits erwähnt, zu jedem Bildpunkt der ersten Aufnahme, entlang der Epipolarlinie nach dem korrespondierenden Bildpunkt. Zu diesem Zweck wird um den Bildpunkt der linken Aufnahme ein Fenster definiert, welches die benachbarten Bildpunkte enthält. Dieses Fenster wird dann mit den entlang der Epipolarlinie verschobenen Fenster der zweiten Aufnahme verglichen. Das ähnlichste Fenster wird als Korrespondenz identifiziert und der Abstand des Mittelpunkts des Fensters aus der ersten Ansicht zum Mittelpunkt des korrespondierenden Fensters entspricht der Disparität und ist somit ein Maß für die Tiefe. Eine große Disparität impliziert dabei ein Objekt nah an der Kamera. Eine kleine Disparität gehört zu einem Objekt weiter weg von der Kamera.

Wird die rechte Aufnahme beispielsweise als Referenz genommen und in der linken Aufnahme nach Korrespondenzen gesucht, muss nur rechts von der Bildkoordinate (also in $x-$ Richtung) eines Punktes der rechten Aufnahme gesucht werden.

3 Resultate

In diesem Abschnitt soll der implementierte Algorithmus anhand einiger Beispiele getestet werden.

3.1 SAD

In einem ersten Schritt soll als Vergleichmaß zur Korrespondenzsuche die SSum of Squared Differences (SAD)"verwendet werden. Die SAD zwischen den beiden Fenster der rechten und linken Bildes ist definiert als

$$\sum_{(i,j) \in W} |I_R(i, j) - I_L(x + i, y + j)| \quad (3.1)$$

Da die SAD nur einfache Rechenoperationen beinhaltet ist auf der Berechnungsaufwand für die Tiefenkarte geringer als bei der Normalized Cross Correlation (NCC). In Abb. 3.1a sind die Ergebnisse des Algorithmus für das Motorrad dargestellt. Abb. 3.1a wurde anschließend noch mit einem Median-Filter bearbeitet und weitere Inhomogenitäten auszugleichen. Das Ergebnis ist in Abb.3.1b dargestellt.

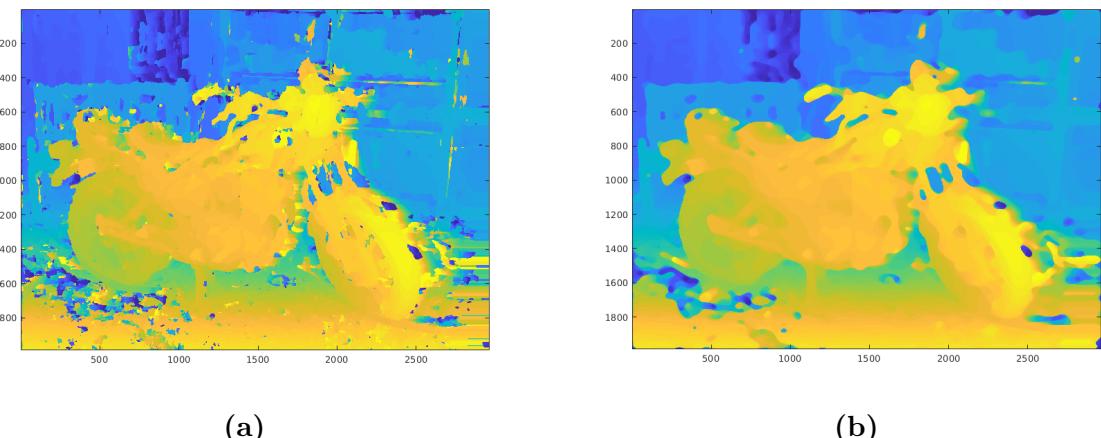


Fig. 3.1. (a) Tiefenkarte Motorrad (b) Tiefenkarte Motorrad mit dem Median Filter bearbeitet

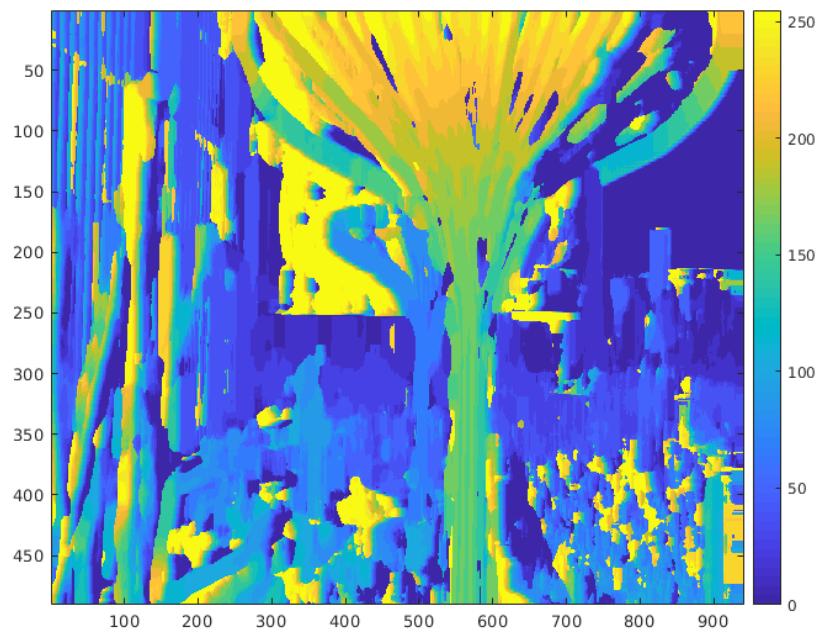


Fig. 3.2. Tiefenkarte Playground ohne Nachbearbeitung

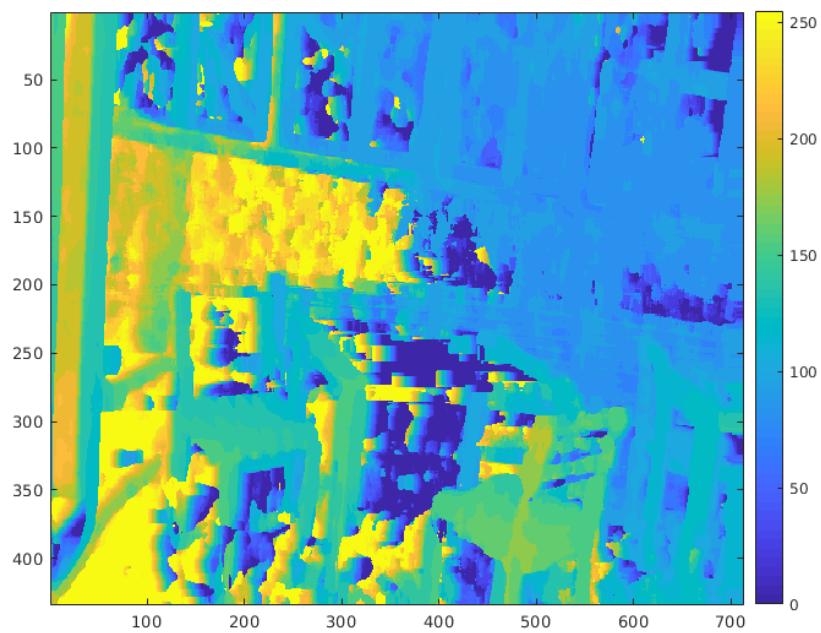
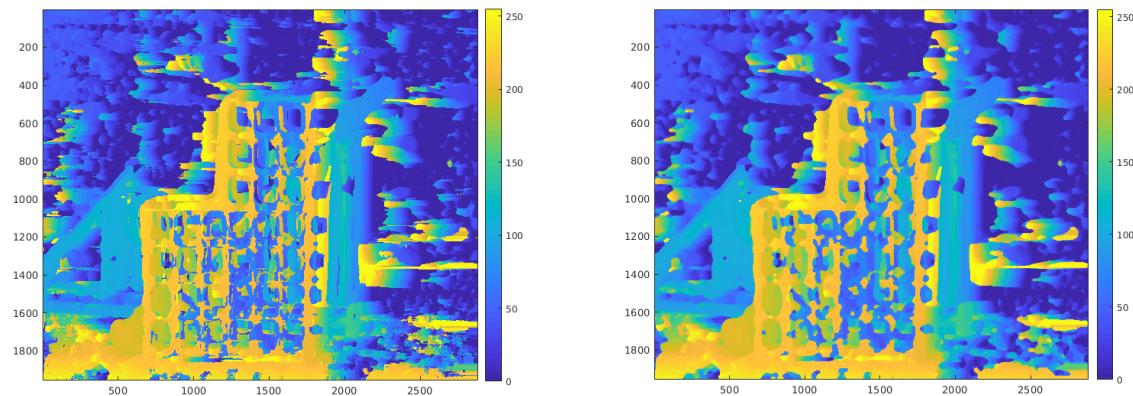


Fig. 3.3. Tiefenkarte Terrace ohne Nachbearbeitung



(a) Tiefenkarte Sword ohne Nachbearbeitung (b) Tiefenkarte Sword mit Nachbearbeitung

Fig. 3.4. Tiefenkarte Sword

3.2 NCC

4 Unitests

In den Unitests werden drei verschiedene Testcases geprüft.

1. Zu erst wird getestet, ob wirklich keine Toolboxen und nur standard MATLAB-Funktionen benutzt werden. Dafür wird eine Funktion geschrieben, die unsere drei Dateien (challenge.m, disparity.m und verify_dmap.m) überprüft. Wenn es mehr als eine Toolbox gibt (MATLAB wird als Standard aufgelistet), wird eine "1" zurückgegeben, ansonsten "0". Bei einer "1" wird der Test als nicht erfolgreich betrachtet.
2. Zusätzlich sollten die Variablen in "Challenge.m" überprüft werden. Die "Challenge.mat"-Datei wird geöffnet, welche die Variablen in "Challenge.m" beinhaltet und überprüft ob diese größer Null und nicht leer sind.
3. Am Ende wird unsere PSNR-Berechnung geprüft, sodass das Ergebnis innerhalb einer Toleranz (10^{-6}) im Vergleich mit der inb der Image Processing Toolbox ist.

Das Resultat des Unitests wird im Command Window gezeigt.

5 GUI

Die grafische Benutzeroberfläche wurde mithilfe der Matlab-Funktion guide erstellt. Diese Funktion wird durch die Eingabe des gleichnamigen Kommandos guide ausgeführt und öffnet eine Benutzeroberfläche, mit der zuerst das Layout der späteren GUI erstellt werden kann. Zum Gestalten stehen hierbei verschiedene vorgefertigte Funktionsblöcke bereit, die eine Ein- und Ausgabe und somit eine Kommunikation mit dem späteren Nutzer ermöglichen. Für diese Anwendung wurden die Bausteine Push Button, Static Text und Axes eingesetzt. Die Static Textboxes und Axes sind dabei reine Output-Fenster, die dem Nutzer das aktuelle Verzeichnis, die darin enthaltenen Bilder, sowie nach der Berechnung auch die Disparitymap und die Ergebnisse für R, T und p anzeigen. Des Weiteren gibt es eine Textbox, die nützliche Hinweise darstellt, wenn das ausgewählte Verzeichnis zum Beispiel keine Bilder mit den Namen Im0 und Im1 enthält und eine weitere, die angibt, ob die Berechnung der Disparitymap erfolgreich war oder nicht. Die Eingabe beschränkt sich auf fünf Push Buttons. Der erste Push Button ist dafür zuständig, eine übersichtliche grafische Benutzeroberfläche zum Suchen und Einlesen des gewünschten Pfads zu öffnen, während mit dem zweiten im Anschluss eine Berechnung der Disparitymap ausgelöst werden kann. Die letzten drei Push-Buttons liegen jeweils unter den Axes-Blöcken. Wird in diesen ein Bild dargestellt, kann dieses zusätzlich in einem Matlab-Figure-Fenster geöffnet werden. Dort werden die Bilder größer dargestellt, können aber zusätzlich mit dem vollen Funktionsumfang von Figure bearbeitet und betrachtet werden. Die zugrundeliegenden Funktionalitäten werden durch, den Buttons zugehörigen Funktionen realisiert, die nach Aktivierung durch den Benutzer ausgeführt werden. Sie werden individuell implementiert und beinhalten zum Beispiel den Aufruf des challenge.m Skripts zur Berechnung der Disparitymap oder die uigetdir-Funktion mit der das Verzeichnis ausgewählt wird. Das Layout, sowie die daraus entstandene Benutzeroberfläche sind auf den folgenden beiden Bildern zu sehen.

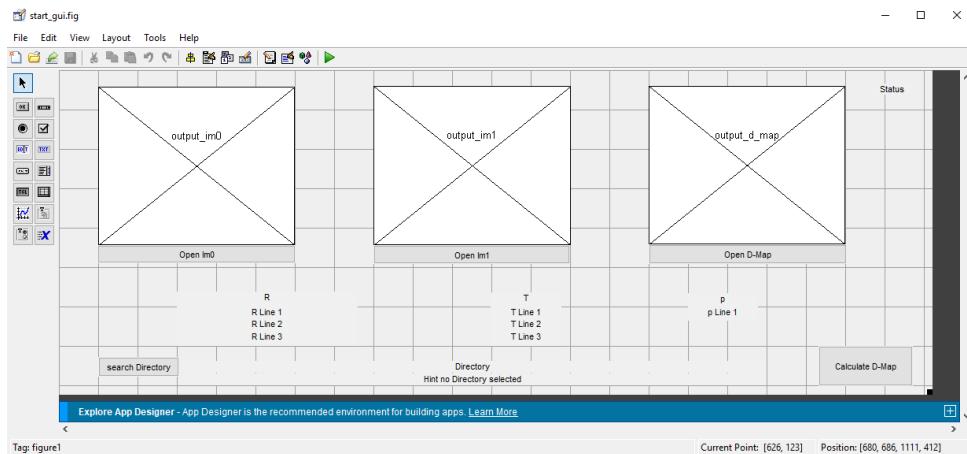


Fig. 5.1. GUI Layout

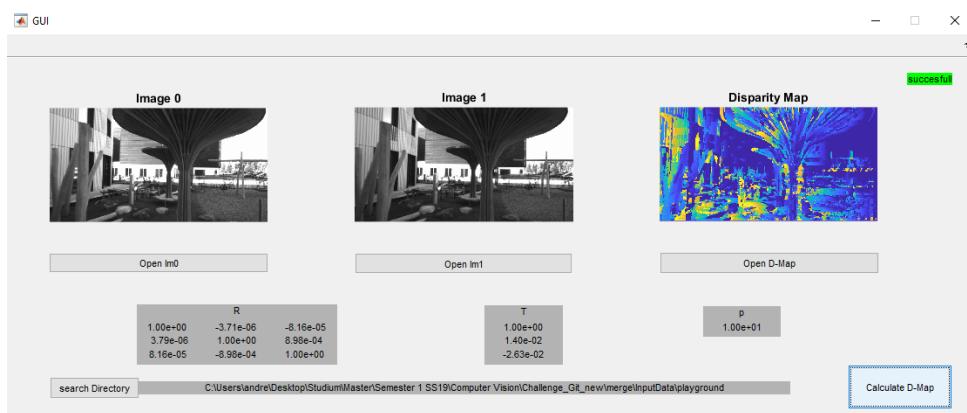


Fig. 5.2. GUI Ausgabe

6 Zusatzfunktionen

6.1 3D Rekonstruktion

Um die 3D - Rekonstruktion

7 Ausblick

Zusammenfassend kann gesagt werden das bei Bildern kleiner größe nicht genügend unterschiedliche bzw Wertvolle Korrespondenzpunktpaare gefunden werden um eine vernünftige Disparity Map zu erstellen. Wie in Kapitel 2.1 gezeigt wurde ist für das Motorrad eine einigermaßen erkennbare Tiefenkarte entstanden, was nur möglich war durch vergleichbar viele und auch weit voneinander liegenden robusten Korrespondenzen.

Daher ist der Ansatz für jeden Pixel in Bild 1 bzw. Block einen Korrespondenz Pixel im Bild 2 zu wählen robuster und auch auf kleinere Bilder anwendbar. Der Nachteil ist hierbei das große Fehler auftreten können und gerade bei homogenen Flächen zu viele Punkten aufeinander fallen wie sehr schön in Kapitel 2.2 zu sehen ist.

Der nächste Schritt wäre also nach einer generierten Tiefenkarte mittels Block Matching eine Segmentierung des original Bildes zu machen um so Fehler in homogenen Flächen zu beheben. Ein Ansatz dafür kann in Abbildung 7.1 gesehen werden. Dort wurde das Bild über einen Schwellwert in ein Binärbild umgewandelt und alle ungefähr zusammenhängende Objekte auf "1" gesetzt.

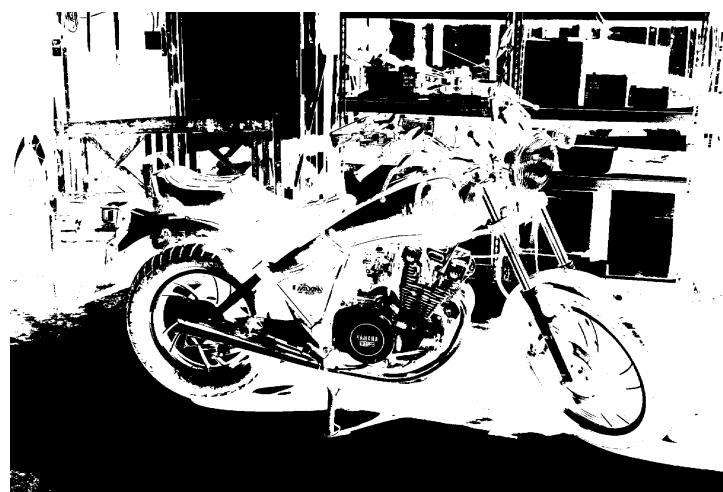


Fig. 7.1. Ansatz einer Segmentierung

Ein weiteren Ansatz den wir noch angefangen haben ist es den Harrisdetektor so um

zuschreiben das er auch Kanten detektiert, dafür haben wir analog zu einem großen Schwellwert für Ecken, einen negativen Schwellwert für "H" von $-5 \cdot 10^{-5}$ gewählt. Damit werden die Korrespondenzen um ein Vielfaches gesteigert, sie sind dadurch so vielzählig das zum berechnen der SVG deutlich mehr Arbeitsspeicher benötigt werden würde. Deshalb müsste man sich überlegen ob man auch hier automatisch erkennen kann durch welche Merkmalspunkte wichtige Informationen über die Objekte im Bild gegeben sind und danach aussortieren. Ein vergleich der Merkmalspunkte des Harrisdetektors so wie wir ihn in den Hausaufgaben implementieren sollten (nur Ecken) ist in Abb. 7.2 und mit zusätzlicher Kanten Detektion in Abb. 7.3 zu sehen.

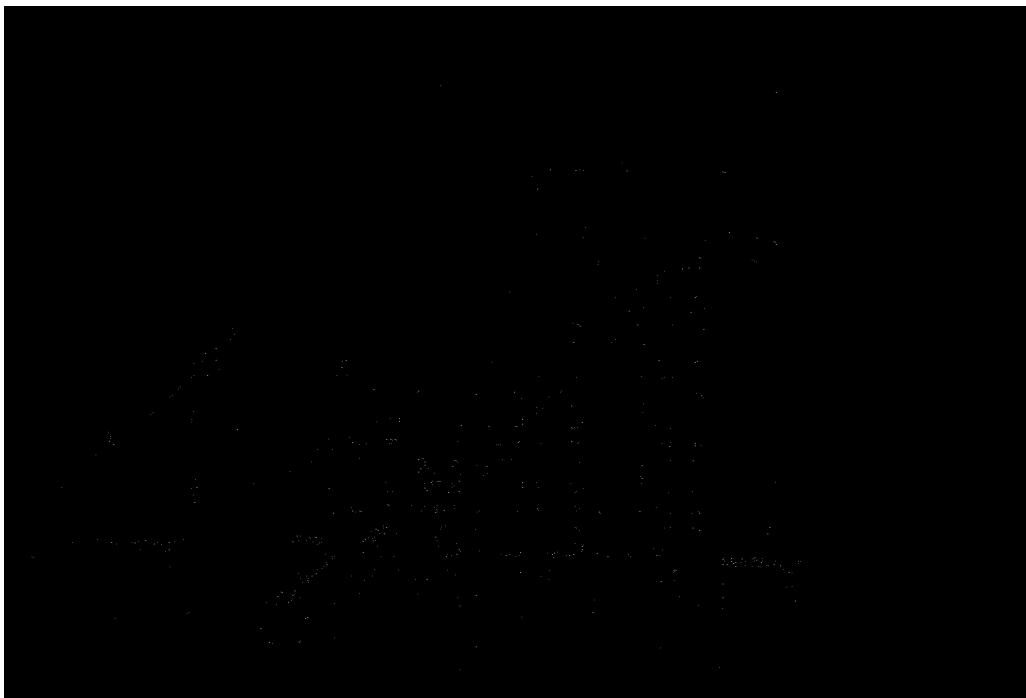


Fig. 7.2. Merkmale von Sword mit Ecken Detektor

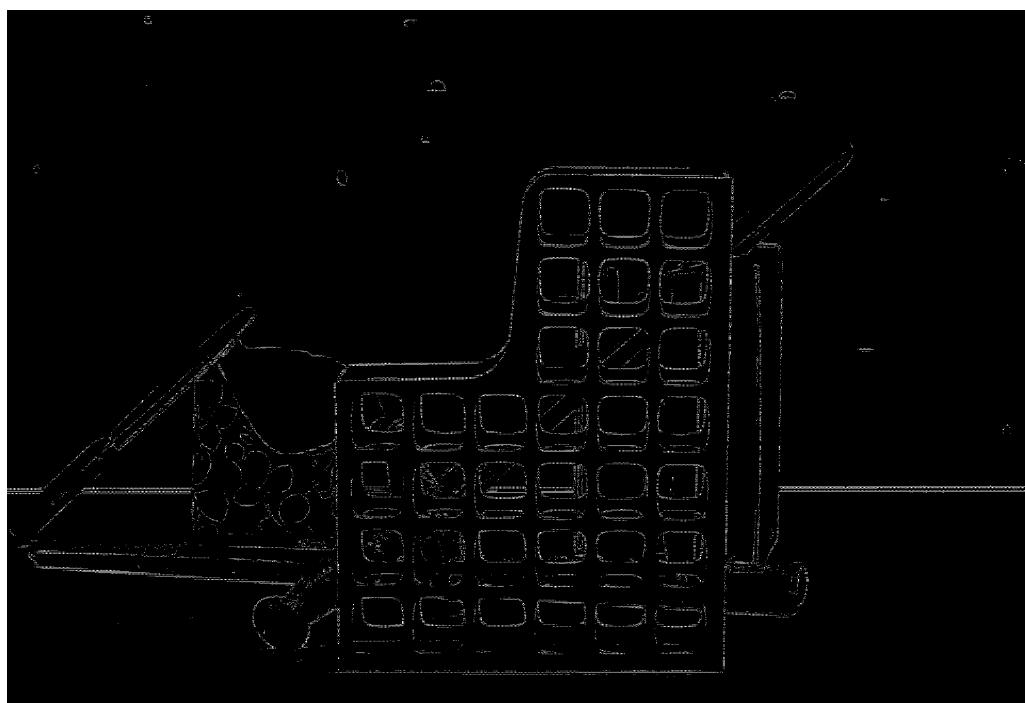


Fig. 7.3. Merkmale von Sword mit Ecken und Kanten Detektor

Abbildungsverzeichnis

1.1	Ablauf des Programms	1
2.1	Flowchart für die Berechnung von Tiefenwerten für Merkmalspunkte	1
2.2	Tiefenkarte des Motorrads mit (a) 1425 und (b) 4194 robusten Korrespondenzen	3
2.3	Playground 643 robuste Korrespondenzen	3
2.4	Sword 46 robuste Korrespondenzen	4
2.5	Terrace 617 robuste Korrespondenzen	4
2.6	Epiloare Einschränkung [1]	5
3.1	(a) Tiefenkarte Motorrad (b) Tiefenkarte Motorrad mit dem Median Filter bearbeitet	1
3.2	Tiefenkarte Playground ohne Nachbearbeitung	2
3.3	Tiefenkarte Terrace ohne Nachbearbeitung	2
3.4	Tiefenkarte Sword	3
5.1	GUI Layout	2
5.2	GUI Ausgabe	2
7.1	Ansatz einer Segmentierung	2
7.2	Merkmale von Sword mit Ecken Detektor	3
7.3	Merkmale von Sword mit Ecken und Kanten Detektor	4

Literatur

- [1] *Epipolar Geometry Kernel Description.* https://en.wikipedia.org/wiki/Epipolar_geometry. Accessed: 2019-07-17 (siehe S. 5).
- [2] Chris McCormick. „Stereo Vision“. In: <http://mccormickml.com/2014/01/10/stereo-vision-tutorial-part-i/> (Jan. 2014) (siehe S. 5).