



Lehrstuhl für
Datenverarbeitung

Technische Universität München **TUM**

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	2
2.1 Epipolareometrie	2
2.2 Block Matching	5
3 Resultate	8
3.1 Sum of Absolute Differences (SAD)	8
3.2 Normalized Cross Correlation (NCC)	11
3.3 Rotationsmatrizen und Translationsvektoren	14
3.4 Peak-Signal-to-Noise-Ratio (PSNR)	16
4 Unitests	17
5 GUI	18
6 Zusatzfunktionen	20
6.1 3D Rekonstruktion	20
7 Ausblick	21

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.

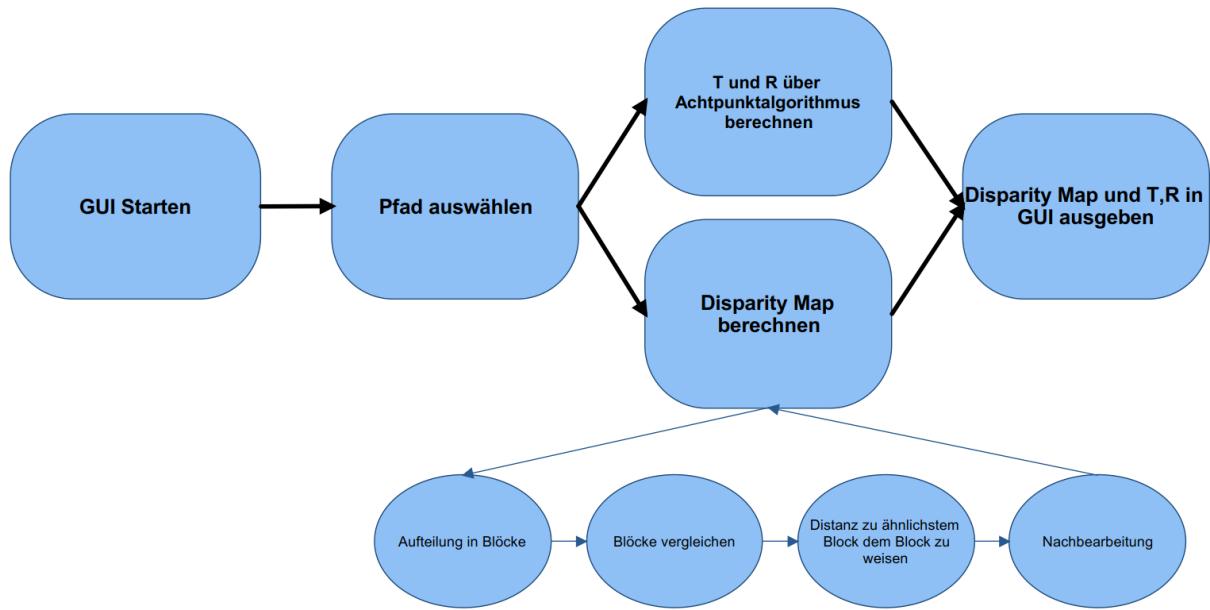


Fig. 1.1. Ablauf des Programms

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.

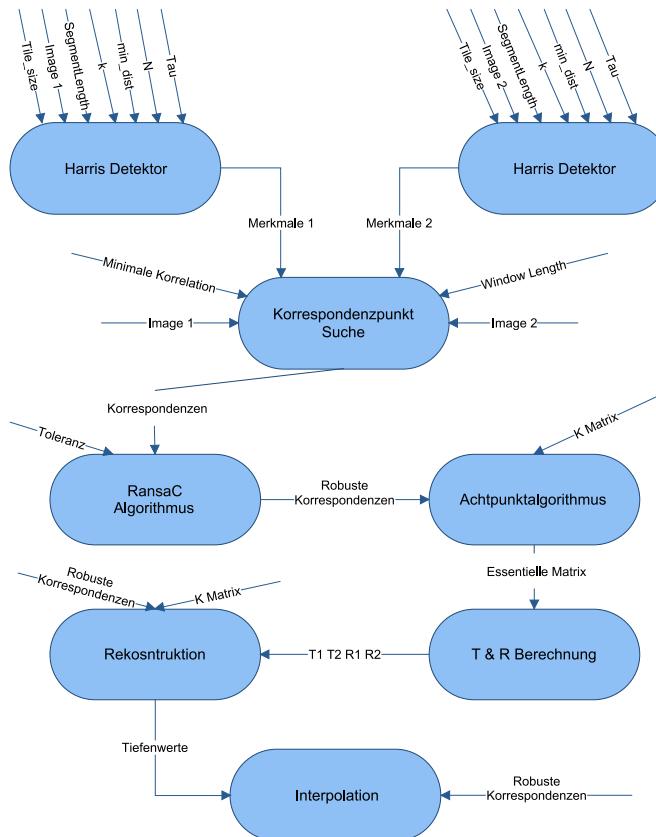
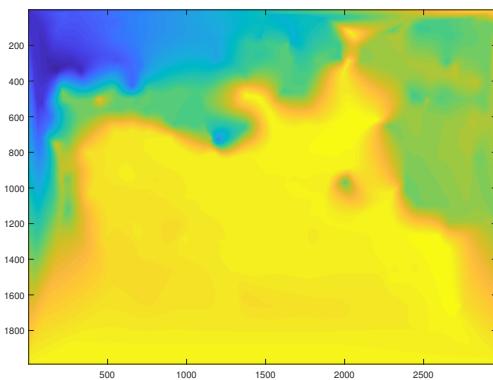


Fig. 2.1. Flowchart für die Berechnung von Tiefenwerten für Merkmalspunkte

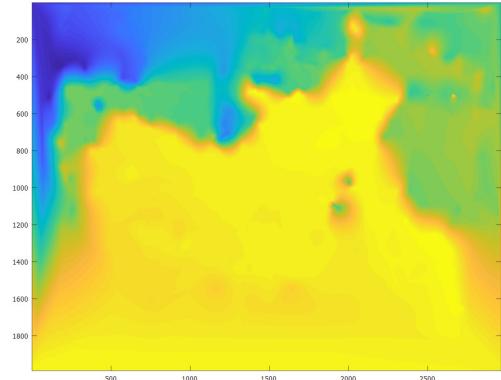
Über den Harrisdetektor werden Ecken und Kanten erkannt und als Merkmalspunkte verwendet. Aus diesen Merkmalspunkten werden robuste Korrespondenzen zwischen den beiden Kameraansichten gesucht. Nur zu den robusten Korrespondenzpunkten kann das Tiefenverhältnis berechnet werden. Um möglichst viele Tiefenwerte (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 robusten 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 robusten Korrespondenzen ihre Tiefenwerte zu und interpolieren zwischen diesen Punkten. Um ein Bild in Größe des Originalbildes zu erhalten, suchen wir die am nächsten liegenden Korre-

spondenzpunkte 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 $N = 50$ &
 $tile_size = [200 \ 200]$

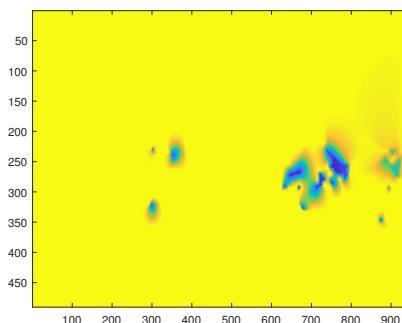


(b) Tiefenkarte des Motorrads mit $N = 300$ &
 $tile_size = [100 \ 100]$

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



(a) Playground Grau



(b) Tiefenkarte Playground



(c) Ground Truth Playground

Fig. 2.3. Playground 643 robuste Korrespondenzen

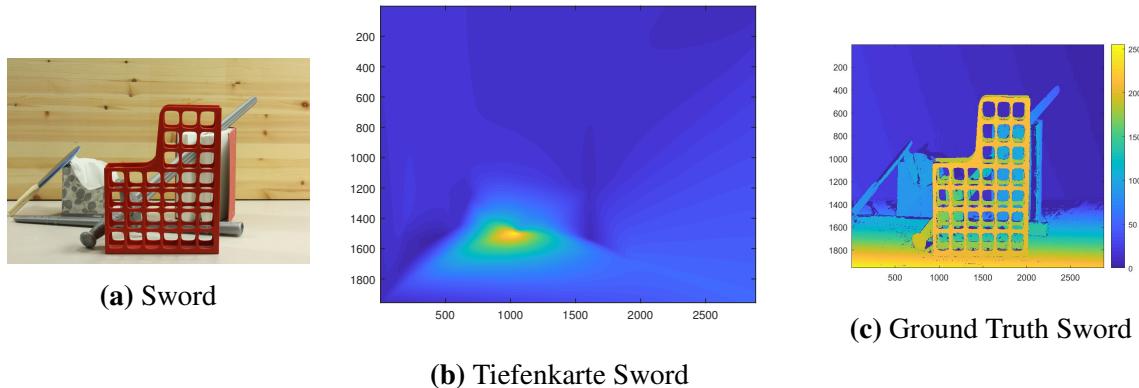


Fig. 2.4. Sword 46 robuste Korrespondenzen

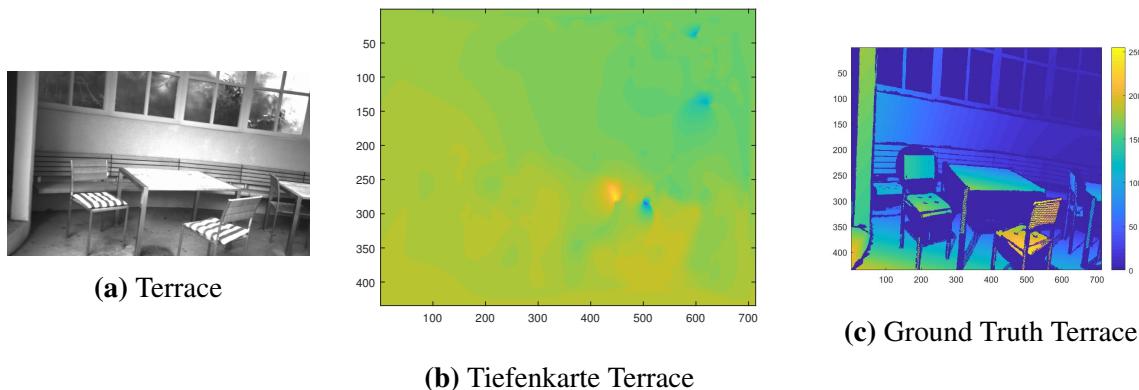


Fig. 2.5. Terrace 617 robuste Korrespondenzen

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 verlaufen. Der Suchraum für den Korrespondenzpunkt in der zweiten Aufnahme zum Bildpunkt der ersten Aufnahme kann entsprechend entlang einer horizontalen Linie stattfinden. Diese horizontale Linie besitzt

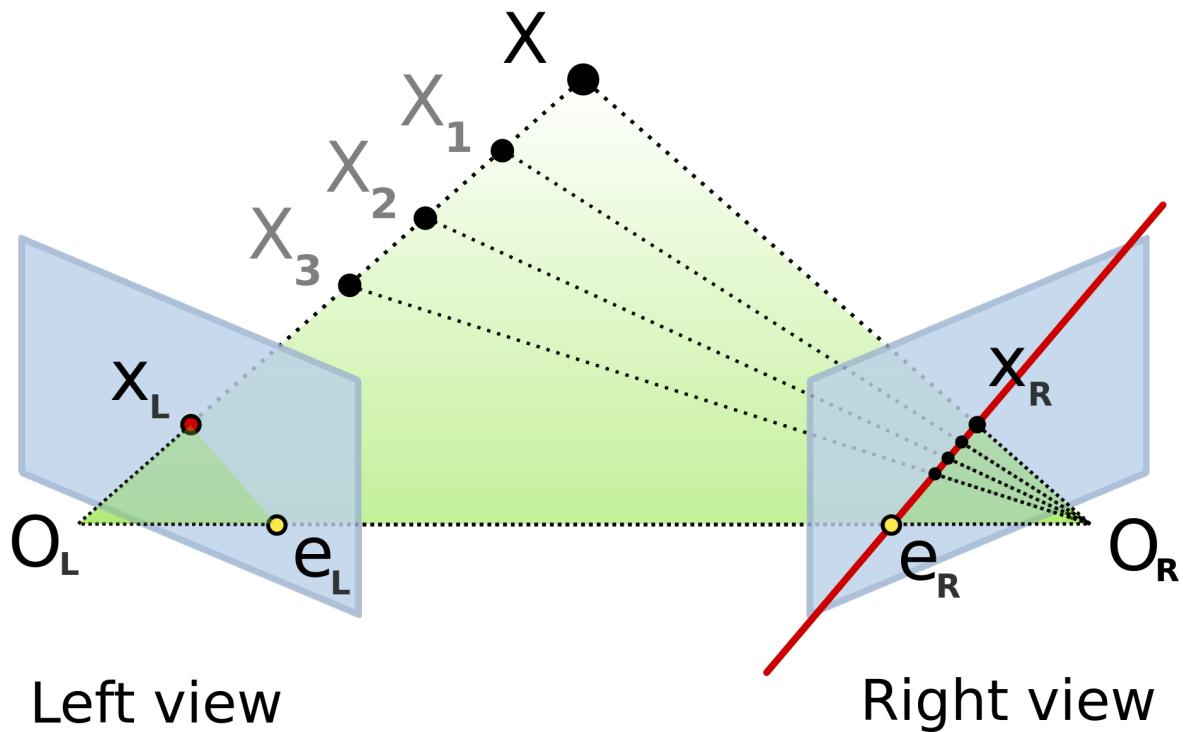


Fig. 2.6. Epiloare Einschränkung [1]

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.

Um den Rechenaufwand zu reduzieren, wird abhängig von der Größe der eingelesenen Datei nicht immer für jedes Pixel die Disparität ermittelt. Dazu wird das Bild in symmetrische Blöcke aufgeteilt einer vordefinierten Größe (z.B. 2×2). Dann wird zu diesem Block der ähnlichste Block

im anderen Bild gesucht. Allen Pixel des Referenzblocks wird dann derselbe Wert gegebenen. Dies reduziert den Rechenaufwand erheblich. Um bei kleinen Bilder nicht die Auflösung der Tiefenkarte zu verringern wird hier von einer Unterteilung der Bilder in Blöcke abgesehen.

3 Resultate

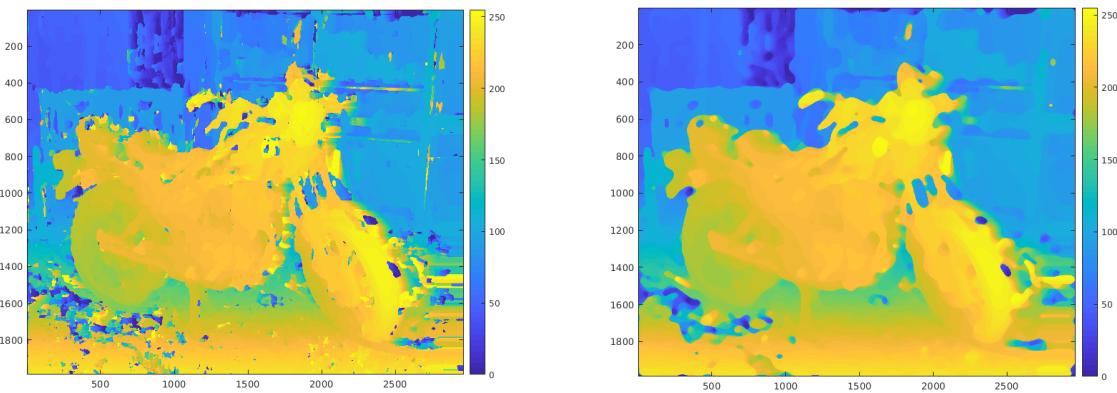
In diesem Abschnitt soll der implementierte Algorithmus anhand einiger Beispiele getestet werden.

3.1 Sum of Absolute Differences (SAD)

In einem ersten Schritt wird als Vergleichmaß zur Korrespondenzsuche die "Sum of Absolute Differences (SAD)" verwendet werden. Die SAD zwischen zwei Fenstern des 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. 3.1 sind die Ergebnisse des Algorithmus für das Motorcycle-Bild dargestellt. Abb. 3.1 wurde anschließend noch mit einem selbst implementierten Median-Filter bearbeitet, um weitere Inhomogenitäten auszugleichen. Das Ergebnis ist in Abb. 3.1b dargestellt.



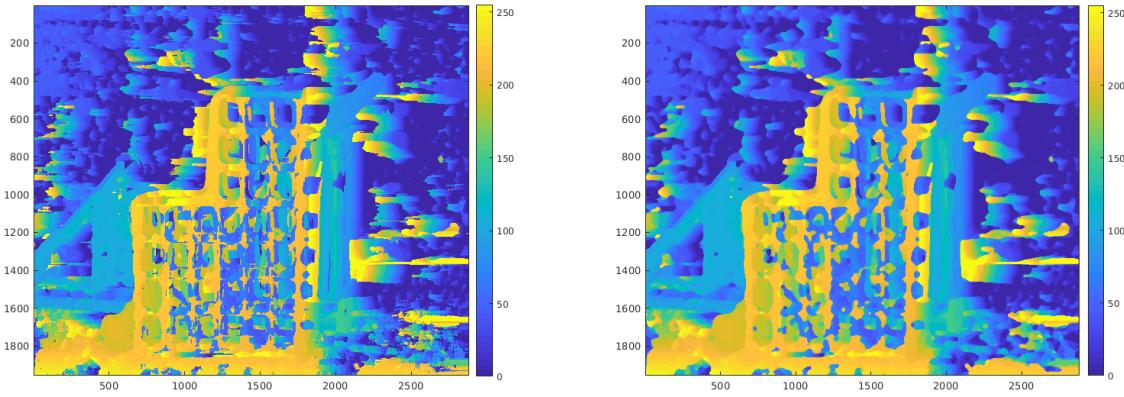
(a) Tiefenkarte Motorcycle unbearbeitet

(b) Tiefenkarte Motorcycle bearbeitet

Fig. 3.1. Tiefenkarten Motorcycle mit SAD als Vergleichsmaß

In Abb. 3.2 sind die Ergebnisse für das Swoord-Stereoset dargestellt. In Abb. 3.2a ohne Nachbe-

arbeitung und in Abb. 3.2b ist das gefilterte Ergebnise dargestellt. Die Größe des Filters wurde dabei an die Bildgröße angepasst. Die Ergebnisse für das Sword-Steroset sind merklich schlechter als beim Motorcycle-Set.



(a) Tiefenkarte Sword ohne Nachbearbeitung

(b) Tiefenkarte Sword mit Nachbearbeitung

Fig. 3.2. Tiefenkarte Sword mit SAD als Vergleichsmaß

Die Abb. 3.3 und Abb. 3.4 sind die Ergebnisse für das Playground und Terrace-Set abgebildet. Diese sind ebenfalls wenig zufriedenstellend. Sie beiinhalten große Bereiche in denen eine falsche Disparität erkannt wurde. Diese Tatsache haben wir zum Anlass genommen das Vergleichsmaß zu ändern.

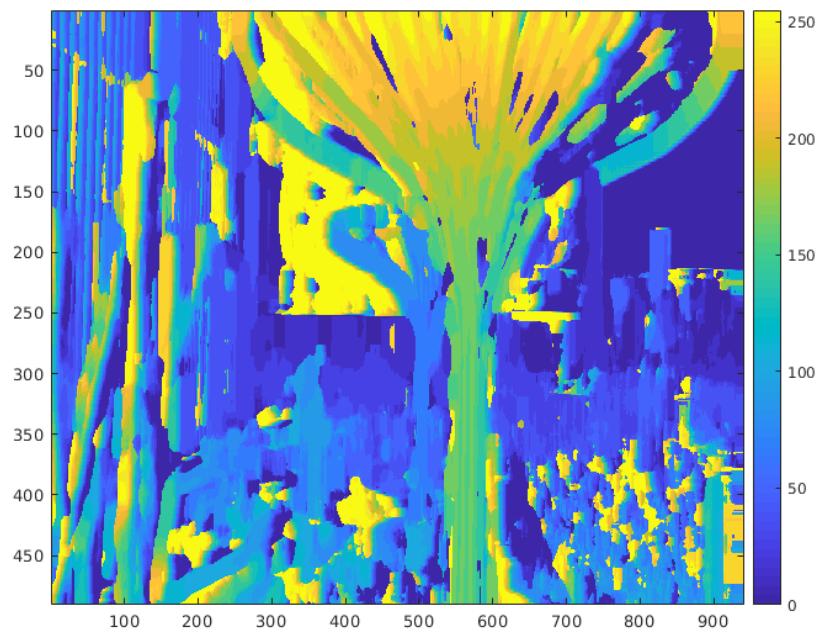


Fig. 3.3. Tiefenkarte Playground mit SAD als Vergleichsmaß und ohne Nachbearbeitung

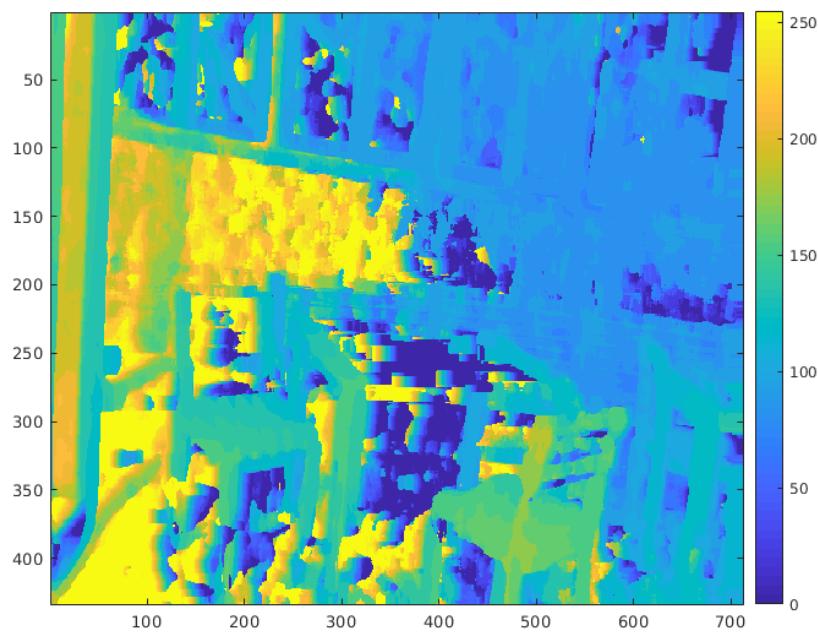


Fig. 3.4. Tiefenkarte Terrace mit SAD als Vergleichsmaß und ohne Nachbearbeitung

3.2 Normalized Cross Correlation (NCC)

Bei der Normalized Cross Correlation handelt es sich, um ein Vergleichsmaß von größerem Berechnungsaufwand als die SAD. Dafür ist dieses Vergleichsmaß invariant gegenüber Intensitätsänderung. Die NCC ist vergleichbar mit dem Vektorprodukt. Bei großer Korrelation berechnet sich die NCC zu 1, bei völliger Unkorreliertheit zu 0. Die Berechnung der NCC ist gegeben durch

$$NCC = \frac{1}{N-1} \text{tr} \left(\frac{(W - \bar{W})^T}{\sigma(W)} \frac{(V - \bar{V})}{\sigma(V)} \right) \quad (3.2)$$

Die Ergebnisse für das Motorrad-Set sind in Abb. 3.5 dargestellt. Es werden jeweils die ungefilterten und gefilterten Ergebnisse dargestellt. Insgesamt lässt sich eine deutliche Verbesserung gegenüber den Ergebnissen aus Kapitel 3.1 feststellen. Durch die Filter kann ein großer Teil des Rauschens entfernt werden. Einige größerer Artefakte lassen sich nicht entfernen. Dies liegt zum Teil daran, dass es zu Okklusionen kommt. Also Teile des linken Bildes sich im rechten Bild nicht finden lassen, da sie von weiter vorne liegenden Objekten im linken Bild verdeckt werden. Im rechten Bild sind durch die Translation in Bezug zu der ersten Aufnahme diese Objekte nun sichtbar.

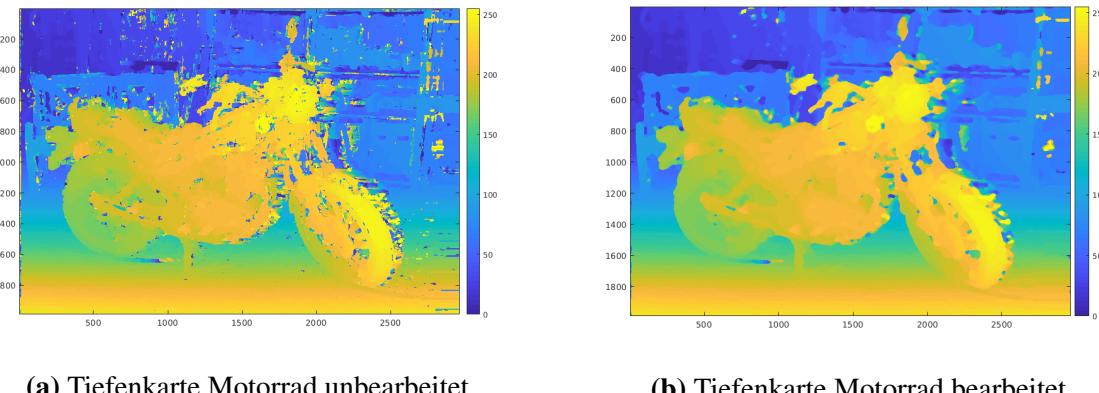
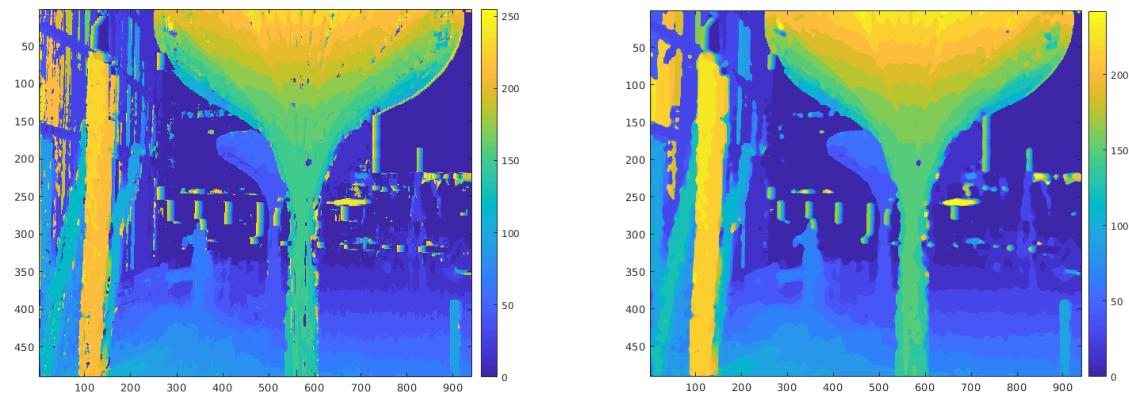
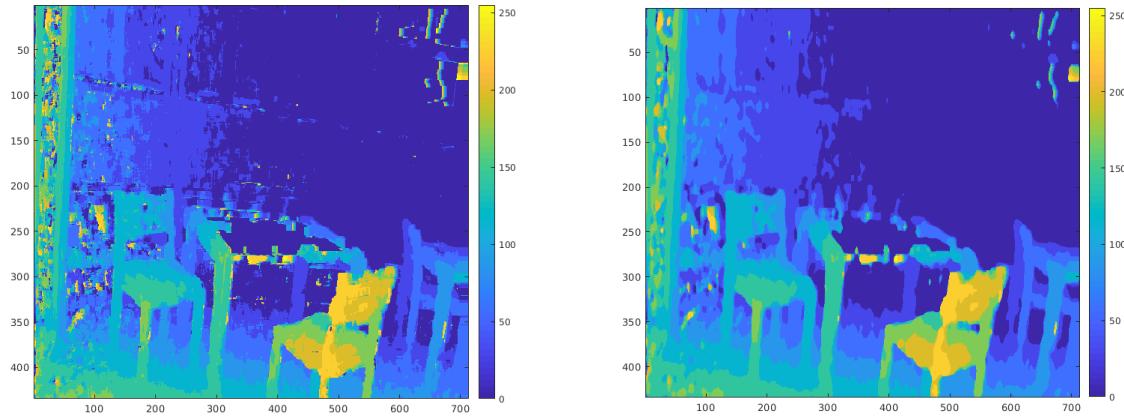


Fig. 3.5. Tiefenkarten Motorrad mit NCC als Vergleichsmaß



(a) Tiefenkarte Playground unbearbeitet

(b) Tiefenkarte Playground bearbeitet

Fig. 3.6. Tiefenkarten Playground mit NCC als Vergleichsmaß

(a) Tiefenkarte Terrace unbearbeitet

(b) Tiefenkarte Terrace bearbeitet

Fig. 3.7. Tiefenkarten Terrace mit NCC als Vergleichsmaß

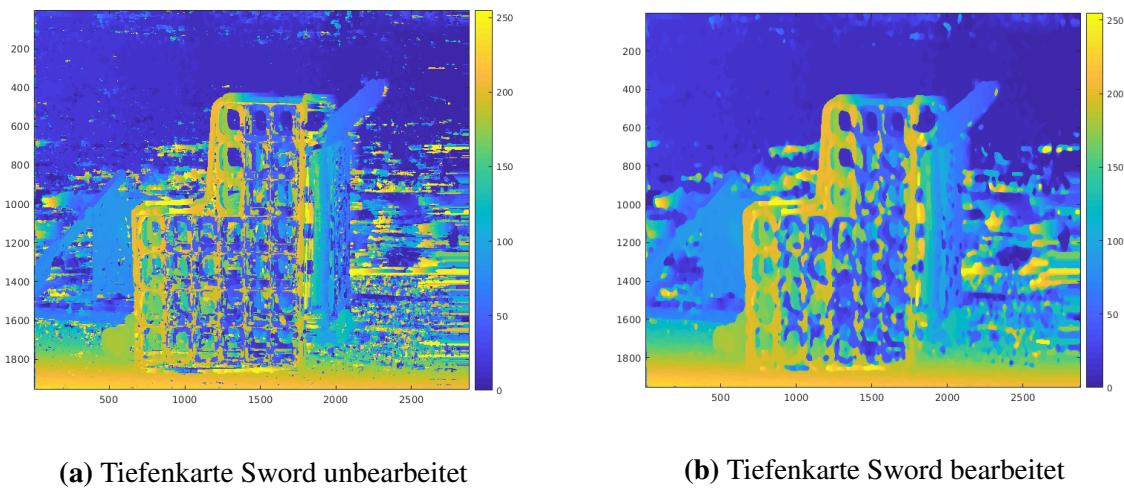


Fig. 3.8. Tiefenkarten Sword mit NCC als Vergleichsmaß

3.3 Rotationsmatrizen und Translationsvektoren

Die Berechnung von Rotationsmatrix und Translationsvektor der korrespondierenden euklidischen Bewegung zur jeweils gegebenen Stereoaufnahme erfolgt mit Hilfe der Hausaufgaben. Es können hier zwei Annahmen getroffen werden:

1. Der Translationsvektor sollte, ohne Normierung auf die Baseline der Kameras, in der ersten Komponente ungefähr -1 sein.
2. Die Rotationsmatrix sollte ungefähr der Einheitsmatrix entsprechen.

Dies liegt daran, dass alle Stereoaufnahmen nur mittels einer Translation der Kamera aufgenommen wurden und ohne Rotation. Für die gegebenen Test-Aufnahmen geben sich folgende Werte, die bereits auf die Baseline der Kameras normiert sind und die Einheit Meter besitzen

$$R_{Motorcycle} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$T_{Motorcycle} = \begin{bmatrix} -0.193 \\ 0.0 \\ 0.0 \end{bmatrix}$$

$$R_{Playground} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$T_{Playground} = \begin{bmatrix} -0.06 \\ 0 \\ 0 \end{bmatrix}$$

$$R_{Sword} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$T_{Sword} = \begin{bmatrix} -0.62 \\ 0.0 \\ 0.0 \end{bmatrix}$$

$$R_{Terrace} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$T_{Terrace} = \begin{bmatrix} -0.06 \\ 0.0 \\ 0.0 \end{bmatrix}$$

3.4 Peak-Signal-to-Noise-Ratio (PSNR)

Der von uns implementierte Algorithmus berechnet jeweils für die Stereoaufnahmen die rechte Tiefenkarte. Im gegebenen Datensatz ist leider nur die Tiefenkarte für das linke Bild gegeben. Im Internet konnte die rechte Tiefenkarte von Motorcycle und Sword ermittelt werden. Diese haben allerdings andere Größen, als die Bilder selber. Die errechneten Werte für die PSNR dienen deshalb eher als grobe Referenz, nicht als valides Bewertungsmaß. Für Playground und Terrace konnte die rechte Referenzkarte leider nicht ausfindig gemacht werden. Trotzdem wurde die PSNR mit der linken Referenzkarte ermittelt

Die PSNR - Werte lauten

1. Motorcycle
 - unbearbeitet: ~16 dB
 - bearbeitet: ~20 dB
2. Sword
 - unbearbeitet: ~12 dB
 - bearbeitet: ~15 dB
3. Playground
 - unbearbeitet: ~10 dB
 - bearbeitet: ~12 dB
4. Terrace
 - unbearbeitet: ~11 dB
 - bearbeitet: ~12 dB

4 Unitests

In den Unitests werden drei verschiedene Testcases geprüft.

1. Zuerst wird getestet, ob keine Toolboxen sondern nur Standard-MATLAB-Funktionen verwendet werden. Dafür wird eine Funktion geschrieben(*check_toolboxes.m*), die die drei Dateien *challenge.m*, *disparity.m* und *verify_dmap.m* überprüft. Diese Überprüfung wird mit der *check_toolboxes*-Funktion durchgeführt. Als Ergebnis der *check_toolboxes*-Funktion wird true oder "false" zurückgegeben, das mit "false" verglichen wird. Der Test wird bei Übereinstimmung als erfolgreich angezeigt.
2. Als nächstes werden die Variablen in "Challenge.m" überprüft. Die globalen Variablen "members", "mail", "D", "R", "T" werden mit *verifyNotEmpty* getestet, wohingegen "elapsed_time", sowie "p" mit *verifyGreaterThan Null* und "group_number" mit *verifyEqual 59* getestet werden.
3. Zuletzt wird das Ergebnis der eigenen PSNR-Berechnung mit dem Ergebnis der von der Image Processing Toolbox bereitgestellten Funktion übereinstimmt. Dabei wird eine Abweichung von 0,1 toleriert.

Die Ergebnisse der Unitests werden alle im Command Window angezeigt.

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 die Grundfunktionalitäten dieser 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.

Im Anschluss wurden ein paar advanced Funktionalitäten hinzugefügt, so zeigt ein zusätzlicher Image-Slot bestehend aus Axes und Button die 3D Rekonstruktion an. Außerdem gibt es einen weiteren Button, der es ermöglicht, das Testskript im Anschluss zur Disparity Map Berechnung auszuführen, sowie zwei Checkbox Blöcke, mit denen eingestellt werden kann, ob die Disparitätsberechnung mit der NCC oder SAD berechnet werden soll und ob das Ergebnis mit einem Median-Filter bearbeitet werden soll. Diese Einstellungen wirken sich auf die Laufzeit und die späteren Ergebnisse aus. Um den Fortschritt beobachten zu können gibt es zusätzlich

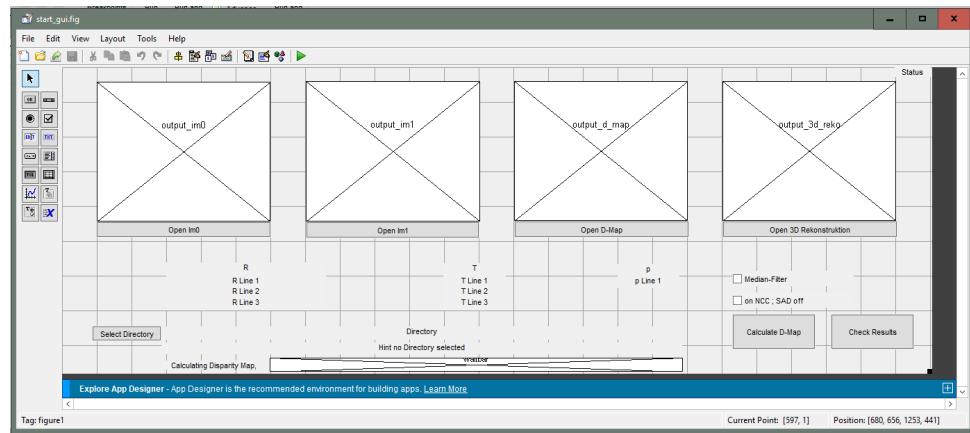


Fig. 5.1. GUI Layout



Fig. 5.2. GUI Ausgabe

einen Fortschrittsbalken, der mithilfe eines letzten Axes dargestellt wird. Das Layout, sowie die daraus entstandene Benutzeroberfläche sind auf den folgenden beiden Bildern zu sehen.

6 Zusatzfunktionen

6.1 3D Rekonstruktion

Um die 3D - Rekonstruktion zu erhalten plotten wir einfach unser Bild mit dem Matlab Befehl "warp" auf die X,Y,Z Koordinaten unserer Tiefenkarte. Der Plot für "Terrace" ist in Abb. 6.1 zu sehen

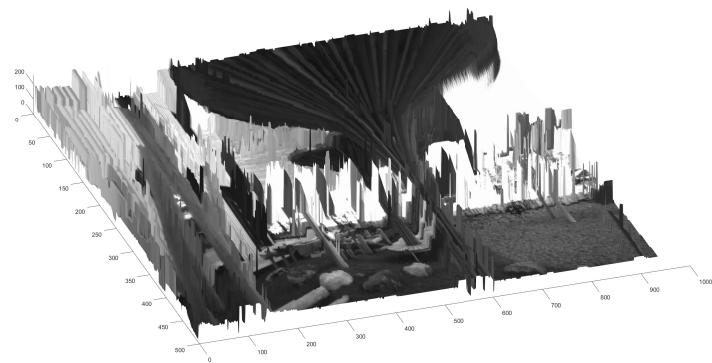


Fig. 6.1. 3D Rekonstruktion

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.

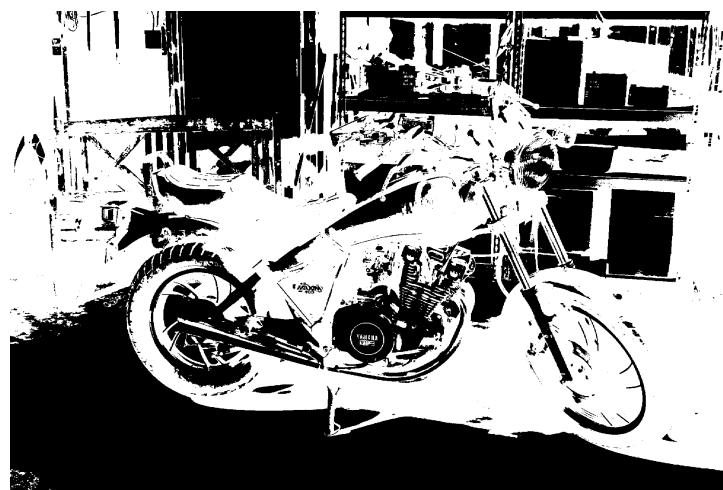
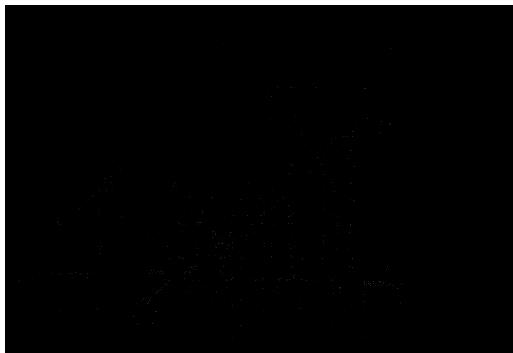


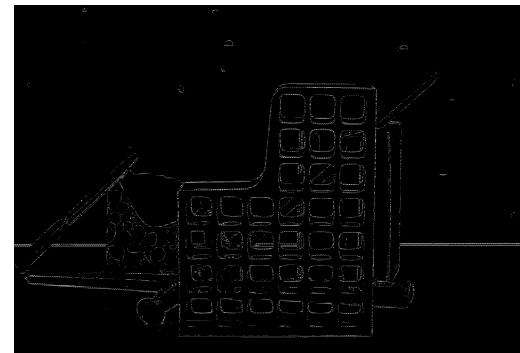
Fig. 7.1. Ansatz einer Segmentierung

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

ben das er auch Kanten detektiert, dafür haben wir analog zu einem großen Schwellwert für Ecken, einen negativen Schwellwert für "H" von $-5 * 10^{-5}$ 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. 7.2a und mit zusätzlicher Kanten Detektion in Abb. 7.2b zu sehen.



(a) Merkmale von Sword mit Ecken
Detektor



(b) Merkmale von Sword mit Ecken und Kanten
Detektor

Fig. 7.2. Tiefenkarte Sword

Abbildungsverzeichnis

1.1	Ablauf des Programms	1
2.1	Flowchart für die Berechnung von Tiefenwerten für Merkmalspunkte	2
2.2	Tiefenkarte des Motorrads mit (a) 1425 und (b) 4194 robusten Korrespondenzen	4
2.3	Playground 643 robuste Korrespondenzen	4
2.4	Sword 46 robuste Korrespondenzen	5
2.5	Terrace 617 robuste Korrespondenzen	5
2.6	Epipoare Einschränkung [1]	6
3.1	Tiefenkarten Motorcycle mit SAD als Vergleichsmaß	8
3.2	Tiefenkarte Sword mit SAD als Vergleichsmaß	9
3.3	Tiefenkarte Playground mit SAD als Vergleichsmaß und ohne Nachbearbeitung	10
3.4	Tiefenkarte Terrace mit SAD als Vergleichsmaß und ohne Nachbearbeitung	10
3.5	Tiefenkarten Motorrad mit NCC als Vergleichsmaß	11
3.6	Tiefenkarten Playground mit NCC als Vergleichsmaß	12
3.7	Tiefenkarten Terrace mit NCC als Vergleichsmaß	12
3.8	Tiefenkarten Sword mit NCC als Vergleichsmaß	13
5.1	GUI Layout	19
5.2	GUI Ausgabe	19
6.1	3D Rekonstruktion	20
7.1	Ansatz einer Segmentierung	21
7.2	Tiefenkarte Sword	22

Literatur

- [1] *Epipolar Geometry Kernel Description.* https://en.wikipedia.org/wiki/Epipolar_geometry. Accessed: 2019-07-17 (siehe S. 6).
- [2] Chris McCormick. „Stereo Vision“. In: <http://mccormickml.com/2014/01/10/stereo-vision-tutorial-part-i/> (Jan. 2014) (siehe S. 6).