Lehrstuhl für
Datenverarbeitung

Dokumentation für Computer Vision Challenge

G26: Chenxi Zou, Verena Feierle, Sarah Minwegen, Ghazal Safian, Weiyi Zhang

12. September 2018

Betreuer:
Stefan Röhrl

n

Inhaltsverzeichnis

Kapitel 1

Aufgabenstellung

Die Aufgabe der Challenge besteht darin, eine dritte virtuelle Ansicht aus einem Stereo-Bildpaar zu generieren. Dabei soll der Blickwinkel der virtuellen Ansicht zwischen den beiden realen Ansichten liegen und durch einen Prozentwert frei bestimmbar sein. Die Challenge wird durch ein Matlab Programm realisiert. Die dritte virtuelle Ansicht soll aus zwei vorgegebenen Farbbildern L1.jpg und R1.jpg bzw. L2.jpg und R2.jpg in einer Funktion free_viewpoint mit einem Parameter p generieren werden. P soll zwischen null und eins wählbar sein. Außerdem soll die Rechenzeit von der Funktion free_viewpoint angezeigt werden und in der Variable elapsed_time gespeichert werden[8].

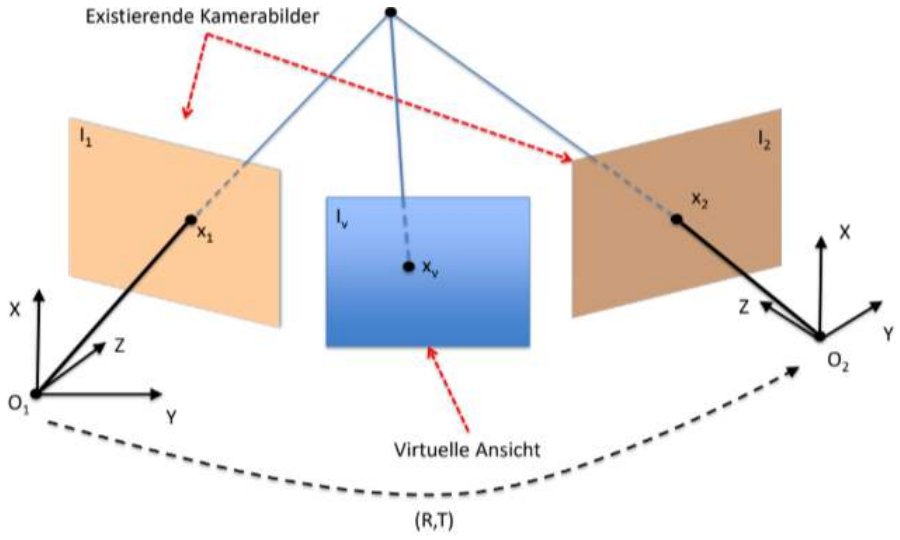

Abbildung 1.1: *Geometrischer Zusammenhang zwischen den realen Bildern und einer künstlich generierten Ansicht [8]*

Kapitel 2

Ansatz und Implementierung

Unser Ansatz, um die virtuelle Ansicht zu erzeugen, folgt der Depth-Image-based Rendering Methode, die drei Hauptschritte beinhaltet. Diese sind das Rektifizieren der Stereo-Bilder, das Erstellen der Disparity Map und zuletzt das Erzeugen der virtuellen Ansicht, wie in 2.1 zu sehen.
Beim Aufrufen des **challenge.m** Skripts werden zuerst die zwei Stereobilder eingelesen, dann die oben genannten Schritte nacheinander durchgeführt und die virtuelle Ansicht generiert und anschließend angezeigt. Die benötigte Laufzeit für die Berechnung des dritten Bildes wird in der Variable elapsed_time gespeichert.

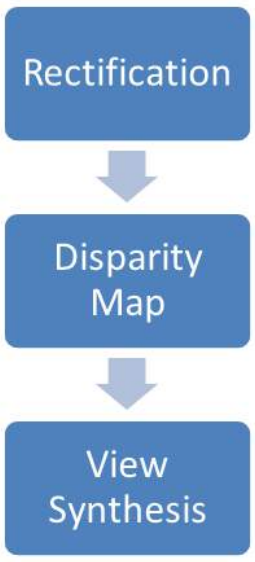

Abbildung 2.1: *Vorgehensweise*

Außerdem wurde für das Projekt eine Benutzeroberfläche implementiert, die dazu dient, die Benutzung zu vereinfachen. Diese kann im Command Window durch *app1* aufgerufen werden.

2.1 Rectification

Wie in 2.1 dargestellt, ist der erste Schritt die Rectification. Ziel dabei ist es, die Epipolarlinien der Stereobilder horizontal auszurichten, damit man zwei Bilder auf der gleichen Ebene erhält. Dafür werden zuerst die Korrespondenzpunkte benötigt.
Im Folgenden sind die wichtigsten Funktionen für die Korrespondenzschätzung kurz genannt und beschrieben.

get_correspondence.m
Hier werden die genauen Korrespondenzpunkte mittels des SIFT-Algorithmus gefunden. Die Implementierung des Codes findet sich hier [6].
Der SIFT Algorithmus lieferte bessere Ergebnisse durch mehr und genauere Korrepondenzpunkte, als bspw. der Harris-Corner-Detektor. Ein Nachteil ist, dass die Laufzeit des SIFT Algorithmus langsamer ist als die des Harris-Corner-Detektors.

lmeds.m
In dieser Funktion wird nun anhand der gefundenen Korrespondenzen die Fundamentalmatrix bestimmt, die später für die Rectification benötigt wird. Diese Funktion basiert auf [7]. Sie ermöglicht die Auswahl zwischen zwei Methoden zur Berechnung der Fundamentalmatrix.

In der Fundenmental_ls.m Funktion wird eine rank-2 Fundamental matrix auf 3*3 anhand des Achtpunktalgorithmus berechnet. Da es nicht nur eine einzige Lösung gibt, wird least sqaures benutzt, um die optimale Lösung zu finden. Die Fundmatrix_nonlin dagegen nutzt die 7-point Methode, wodurch man in unserem Fall einen kleineren Schätzungsfehler erhält und daher diese verwendet wurde.

Wir haben folgende Fundamentalmatrizen erhalten:

$$F1 = \begin{bmatrix} 0 & 0 & 0.0064 \\ 0 & 0 & -0.3898 \\ 0.0013 & 0.3878 & -0.8488 \end{bmatrix} \tag{2.1}$$

für das Bildpaar L1/R1 und

$$F2 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -0.2462 \\ -0.0020 & 0.2483 & -0.9388 \end{bmatrix} \tag{2.2}$$

für das Bildpaar L2/R2.

rectify_images.m

Hier werden durch die berechnete Fundamentalmatrix zunächst die Epipolarlinien und anschließend die Homographiematrizen H1 für das linke und H2 für das rechte Bild berechnet. Damit können die Stereo-Bilder auf die gleiche Ebene transformiert werden und wir erhalten dadurch zwei rektifizierte Bilder. Dies vereinfacht die Berechnung der Disparity Map und der virtuellen Ansicht. Die Impelementierung der Rectification basiert auf [6].

In den folgenden Abbildungen sind die beiden rektifizierten Bilder für das erste und das zweite Bildpaar zu sehen. Zusätzlich stellt die Abbildung 2.6 die rektifizierten Bilder mit den horizontalen Epipolarlinien für das erste Bildpaar dar.

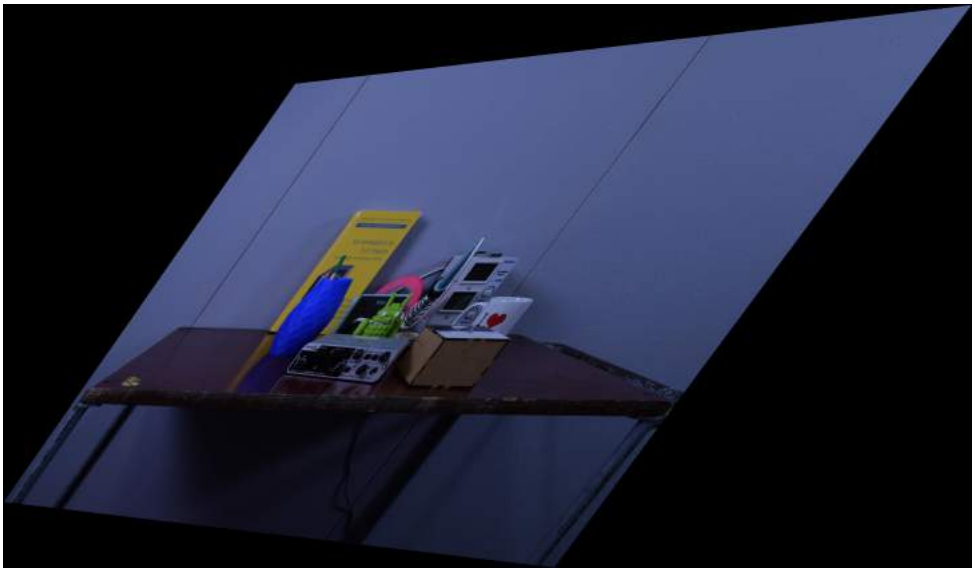

Abbildung 2.2: *Rektifiziertes erstes linkes Bild L1*

Abbildung 2.3: *Rektifiziertes erstes rechtes Bild R1*

Abbildung 2.4: *Rektifiziertes zweites linkes Bild L2*

2.2 Disparity Map

Als nächstes wird aus den rektifizierten Bildern die Tiefeninformation dieser Bilder berechnet. Dafür wird die Disparity Map berechnet, welche die örtliche Differenz der einzelnen Korrespondenzpunkte aus den zwei rektifizierten Bildern beinhaltet.
Dabei wurden zwei unterschiedliche Methoden zur Berechnung der Disparity Map angewendet und getestet.

Erste Methode:
stereomatch.m
Hier wird ein Stereo Matching Algorithmus nach [2] verwendet. Um die Disparity zwischen den beiden Bildern zu erhalten, wird blockweise die "sum of absolute differences (SAD)"

Abbildung 2.5: *Rektifiziertes zweites rechtes Bild R2*

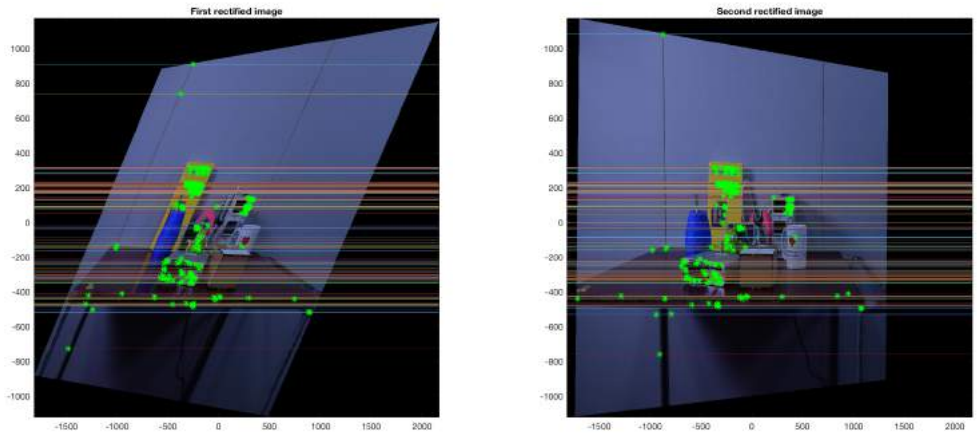

Abbildung 2.6: *Rektifizierte Bilder mit horizontalen Epipolarlinien und Korrespondenzpunkten für das erste Bildpaar*

zwischen den beiden rektifizierten Bildern berechnet. So wird die Ähnlichkeit und somit Disparity zwischen den Bildern berechnet.

Durch das Ändern der Parameter Blockgröße und Disparitybereich erhält man unterschiedlich gute Ergebnisse. Für unseren Fall haben wir uns für eine Blockgröße von 7 und eine Disparity von 254 entschieden, da wir damit das beste Ergebnis erzielen konnten. Durch die große Disparity wurde allerdings die Laufzeit verlängert.

Zweite Methode:

dynamic disparity.m

Der Dynamic Programming Algorithmus [4] vergleicht in jedem Block die benachbarten Pixel spaltenweise. Um die Disparity optimal zu schätzen, wird hier Block Matching als Cost Function benutzt. Im Vergleich zum Stereo Matching Algorithmus war die Laufzeit dieses Algorithmus sehr groß, aber das Ergebnis nicht deutlich besser, weshalb wir uns

gegen den Dynamic Programming Algorithmus entschieden haben.

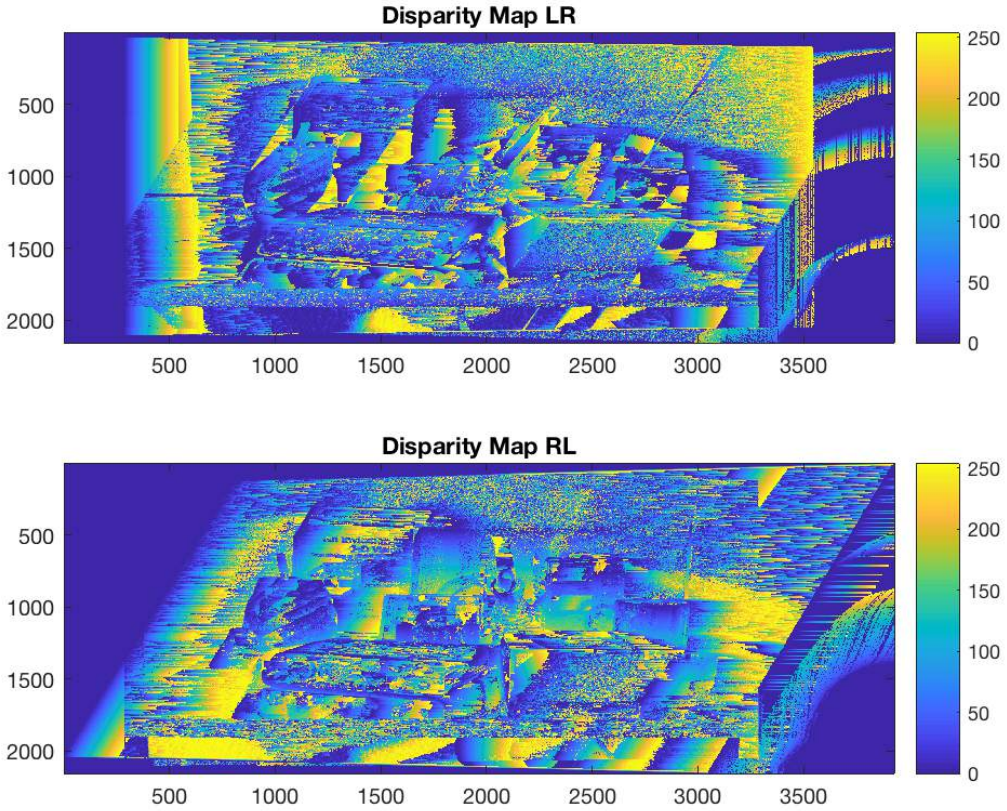

Abbildung 2.7: *Disparity Map mittels Stereo Matching*

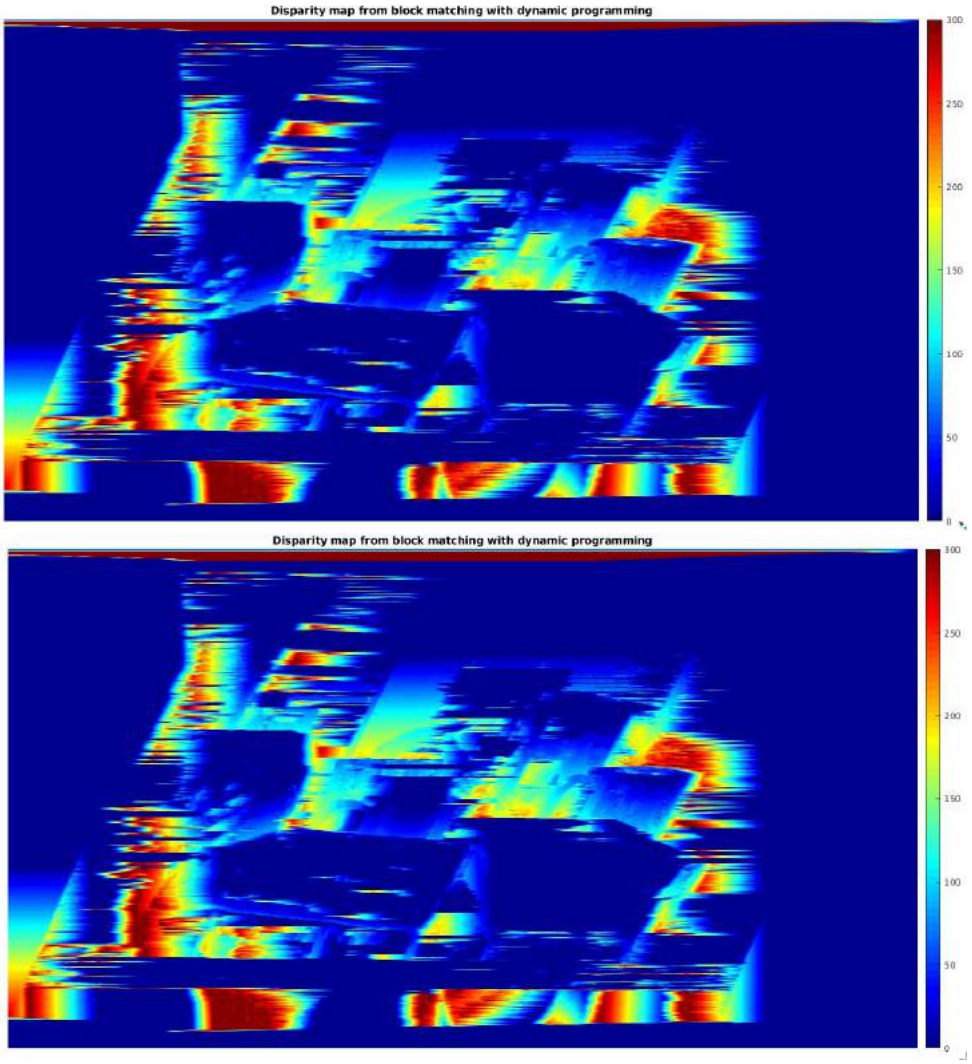

Abbildung 2.8: *Disparity Map mittels Dynamic Programming*

2.3 View Synthesis

2.3.1 Erzeugen der virtuellen Ansicht

Das Erzeugen einer neuen Ansicht anhand des rektifizierten Stereo Bildpaares und deren Disparity Maps basiert auf der "Depth Image-based Rendering and Hierarchial Clustering" Methode, die in [5] vorgestellt wurde. Die Implementierung des von uns angewendeten Algorithmus ist von [1]. Die Umsetzung der Methode besteht aus sechs Hauptfunktionen, die in diesem Abschnitt kurz genannt und beschrieben werden:

removeGhostContour.m

In dieser Funktion werden die Ghost Konturen der beiden rektifizierten Ansichten korrigiert. Ghost Konturen kommen dann zu Stande, wenn die Pixel im Hintergrund des Bildes durch die Farbe des Bildes im Vordergrund kontaminiert sind. In dieser Funktion werden zuerst die Ecken im Bild detektiert. Dafür muss der Canny edge detector auf die Disparity Map des entsprechenden Bildes angewandt werden. Dadurch wird zwischen Pixeln,

die im Bild im Hintergrund sind und Pixeln, die im Vordergrund sind, unterschieden. Anschließend wird ein Median Filter angewendet. Dadurch werden die im Hintergrund detektierten Kanten durch die Werte benachbarter Pixel ersetzt.

initialSynthesize.m
Die dritte Ansicht wird in dieser Funktion erzeugt. Hier werden basierend auf dem 3D Warping Algorithmus in [3] die neue Ansicht und deren Disparity Map berechnet. Bei dieser Methode wird davon ausgegangen, dass das Stereo Bilderpaar rektifiziert ist. Für die dritte Ansicht werden die Pixel in den Referenzbildern in die gewünschte neue Position gemappt.

Type2Fill.m
Die hierarchical clustering Methode, basierend auf [5], wurde in dieser Funktion implementiert.

2.3.2 Füllen der Löcher im erzeugten Bild

Nachdem die dritte Ansicht erzeugt wurde, müssen die Löcher in der virtuellen Ansicht gefüllt werden. Die folgende Funktionen dienen dazu:

colorHoleFill.m
Beim Anwenden dieser Funktion werden die Löcher im RGB-Bild gefüllt. Dabei wird ein Bilateral Filter angewendet.

depthHoleFill.m
Hier wird der Median Filter angewandt, um die Löcher in der Disparity Map zu füllen.

edgeSmooth.m
Für eine realistischere Ansicht wird als letzer Schritt der Median Filter verwendet, welcher die Pixel auf den Kanten glättet.

2.4 Graphische Benutzeroberfläche

Um die virtuelle Sicht anzuzeigen, stellt der User zunächst die gewünschte neue Position für das virtuelle Bild durch den Parameter p ein. Anschließend kann man zwischen den zwei Bildpaaren wählen, für welches das neue virtuelle Bild erzeugt und angezeigt wird.

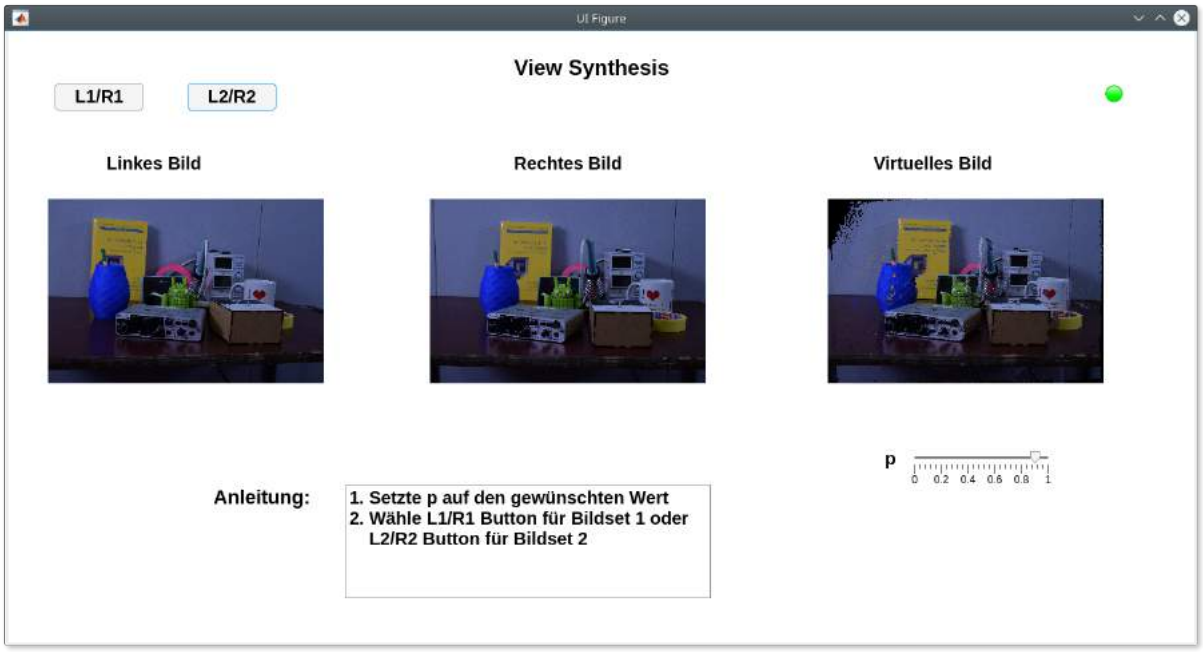

Abbildung 2.9: *GUI*

2.5 Ergebnis

In den folgenden Abbildungen werden die virtuellen Ansichten für L1/R1 und L2/R1 für verschiedene Positionen gezeigt. Dabei betragen die Positionswerte $p = 0.20, 0.45, 0.70, 1$.

Abbildung 2.10: *Virtuelle Ansicht mit p=0.2 für das erste Bildpaar*

Abbildung 2.11: *Virtuelle Ansicht mit p=0.45 für das erste Bildpaar*

Abbildung 2.12: *Virtuelle Ansicht mit p=0.7 für das erste Bildpaar*

Abbildung 2.13: *Virtuelle Ansicht mit p=1 für das erste Bildpaar*

Abbildung 2.14: *Virtuelle Ansicht mit p=0.2 für das zweite Bildpaar*

Abbildung 2.15: *Virtuelle Ansicht mit p=0.45 für das zweite Bildpaar*

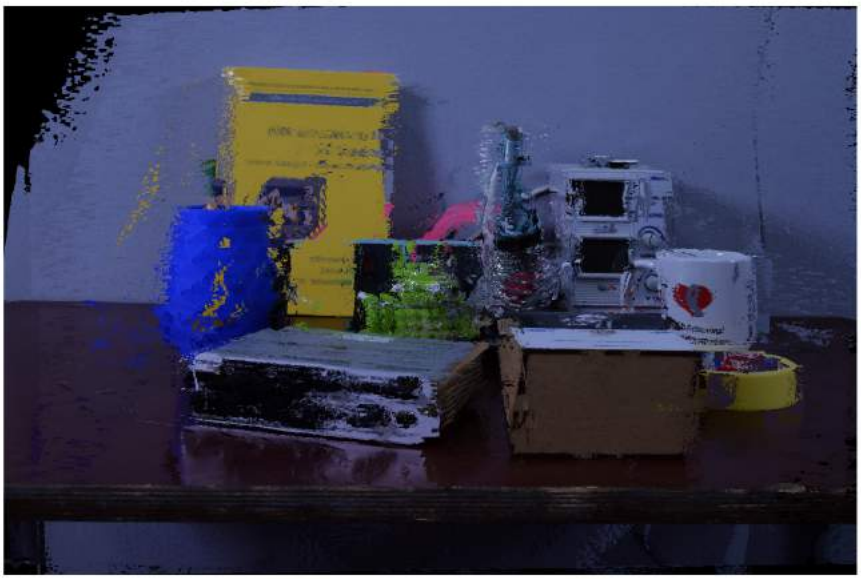

Abbildung 2.16: *Virtuelle Ansicht mit p=0.7 für das zweite Bildpaar*

Abbildung 2.17: *Virtuelle Ansicht mit p=1 für das zweite Bildpaar*

2.6 Laufzeit der Berechnung

Zur Verbesserung der Laufzeit der Berechnung wurden die rektifizierten Bilder auf die halbe Orginalgröße verkleinert. So wurde eine beträchtlich kürzere Laufzeit bei Berechnung der Disparity Map erreicht. Diese beträgt 70.06 s. Allerdings kann an den folgenden Bildern 2.18 - Ansicht mit rektifizierten Bildern auf Originalgröße und 2.19 - Ansicht mit verkleinerten rektifizierten Bildern gesehen werden, dass dies mit einer Qualitätseinbuße verbunden ist. Vergleicht man das erzeugte virtuelle Bild bei dem gewählten Parameter p = 0.2, so ähnelt das Bild bei dem Code mit der verbesserten Laufzeit stärker dem rechten Orginalbild statt wie erwartet dem linken Orginalbild.
Daher wird auf das Verkleinern der Bilder verzichtet, um eine bessere Qualität der Bilder zu erhalten, auch wenn dies mit einer langsameren Laufzeit verbunden ist. Die Laufzeit für das gesamte Berechnen der erzeugten Bildern zu den Ansichten p = 0.2, 0.45, 0.7, 1 beträgt 1062 s.

Abbildung 2.18: *Virtuelle Ansicht mit p = 0.2 für das erste Bildpaar in Originalgröße*

Abbildung 2.19: *Virtuelle Ansicht mit $p = 0.2$ für das erste Bildpaar mit halber Orginalgröße*

Kapitel 3

Schlussfolgerung

Die Visualisierung der Ergebnisse weist noch einige Löcher und Qualitätseinbußen auf. Dies könnte auf die nachfolgend genannten Ursachen zurückzuführen sein:

- Teilweise nicht passende Korrespondenzpunkte resultieren in nicht übereinstimmenden Epipolarlinien. Dies beeinflusst die Rectification und somit auch das Endergebnis.
- Schlechte Lichtverhältnisse der Stereo-Bilder konnten nicht ausgeglichen werden.

Literaturverzeichnis

[1] jidai-code/view-synthesis. https://github.com/jidai-code/View-Synthesis. Accessed: 2018-09-11.

[2] Abbeloos, W. Stereo matching. https://de.mathworks.com/matlabcentral/fileexchange/28522-stereo-matching. Eingesehen am 11.09.2018.

[3] Ahn, I., and Kim, C. A novel depth-based virtual view synthesis method for free viewpoint video. *IEEE Transactions on Broadcasting 59*, 4 (Dec 2013), 614–626.

[4] Perform block matching with dynamic programming. https://inside.mines.edu/~whoff/courses/EENG512/lectures/other/stereo_BlockMatchDynamicProg.m. Eingesehen am 11.09.2018.

[5] Dai, J., and Nguyen, T. View synthesis with hierarchical clustering based occlusion filling. In *2017 IEEE International Conference on Image Processing (ICIP)* (Sept 2017), pp. 1387–1391.

[6] Di, X. Mathworks, sift feature extreaction. https://de.mathworks.com/matlabcentral/fileexchange/50319-sift-feature-extreaction. Eingesehen am 11.09.2018.

[7] Huynh, D. A short tutorial on image rectification. http://staffhome.ecm.uwa.edu.au/~00050673/Tutorials/rectification/. Eingesehen am 11.09.2018.

[8] Klaus Diepold, S. R. Computer vision challenge 2018.