

Computer Vision Challenge

Gruppe 58

Dominik Thoma, Falk Schönfeld, Kathrin Khadra,

Laila Niazy, Zuhra Amiri

12. September 2018

Inhaltsverzeichnis

1 Aufgabe und Rahmenbedingungen	3
2 Bildvorverarbeitung	3
2.1 Generierung der Graubilder	4
2.2 Harris Detektor	4
2.3 Korrespondenzen	6
2.4 RanSaC-Algorithmus	7
2.5 Achtpunktalgorithmus	9
2.6 Rekonstruktion von Rotation und Translation	10
3 Rektifizierung	11
3.1 Rektifizierung der Kameramatrizen	11
3.2 Transformation	12
3.3 Ergebnisse	12
4 Tiefeninformationskarte	13
4.1 Region Based Stereo Matching Algorithmus	13
4.2 Block-Matching mit SAD	14
4.2.1 Summe der absoluten Differenzen	14
4.2.2 Erstellung der Tiefeninformationskarte	15
5 Rendering basierend auf Tiefeninformationen	16
5.1 Rendering mit Inverse Mapping	16
5.2 Rendering mit 3D Warping	18
5.2.1 Entfernen der Schattenkontur	18
5.2.2 3D Warping	18
5.2.3 Fusionieren der Bilder	19
6 Ergebnisse	19
6.1 Ergebnisse mit Inverse Mapping	19
6.2 Ergebnisse mit 3D-Warping	20
7 Graphic User Interface	22

1 Aufgabe und Rahmenbedingungen

Die Computer Vision Challenge ist ein Teil der Master-Vorlesung *Computer Vision* des Studiengangs Elektrotechnik und Informationstechnik der Technischen Universität München. Sie dient dem tieferen Verständnis der Materie, da sich die Studenten vermehrt mit den praktischen Aspekten der behandelten Themen auseinandersetzen. Außerdem kann der Student mit der Abgabe seine Note ohne den direkten Stress einer Prüfungssituation beeinflussen und sein erlerntes Wissen präsentieren.

Das Ziel der Challenge ist es, aus einem Stereo-Bildpaar eine dritte virtuelle Ansicht V zu generieren. Die neue Ansicht soll zwischen den reellen Ansichten liegen. Die genaue Position wird durch einen Eingabeparameter p bestimmt, der den prozentuellen Anteil der Verschiebung vom linken auf das rechte Bild angibt, mit der die virtuelle Ansicht erreicht wird. Der Wert von p liegt zwischen 0 und 1, die jeweils die Maximalpositionen - das linke und das rechte Bild - darstellen. Im Folgenden wird das hier verwendete System anhand des Bildpaars L2 und R2 (in Abbildung 1) vorgestellt. L2 ist dabei die linke Ansicht, R2 die rechte. Der erste Schritt ist die Extraktion von Merkmalen aus beiden Bildern. Aus den Merkmalen werden dann korrespondierende Paare ermittelt und Korrespondenzpunktpaare zusammengeführt. Unter den gefundenen Punktpaaren werden mithilfe des RanSac Algorithmus optimierte Paare gesucht. Aus diesen robusten Korrespondenzpunktpaaren wird die essentielle Matrix errechnet. Aus dieser können die Rotation R sowie die Translation T vom einen ins andere Bild berechnet werden. Mit p können sie skaliert werden, um die Transformation von R2 zu V zu definieren. Mithilfe von R und T können außerdem L2 und R2 rektifiziert werden sowie eine Tiefenkarte erstellt werden. Mit dieser kann Depth Image based Rendering auf die Bilder angewendet werden, um die Stereoaufnahme zu generieren. Die Algorithmen werden im folgenden Kapitel erläutert.



Abbildung 1: Eingabe: Stereofarbbilder L2 und R2

2 Bildvorverarbeitung

Das Bildvorverarbeitung schätzt aus den zwei Stereofarbbildern die essentielle Matrix E sowie die daraus resultierende Rotation R und Translation T der euklidischen Transformation zwischen den Bildern.

Dafür werden der Funktion **findKorrespondenzen.m** beide Bilder und eine Kalibrierungsmatrix übergeben. Die damit ermittelten Korrespondenzen werden in der Funktion **achtpunktalgorithmus.m** verwendet um E schätzen. Im letzten Schritt werden R , T und die Tiefeninformation λ durch die Funktion **euklidischeTransformation.m** berechnet.

Im Folgenden werden die einzelnen Schritte genauer dargestellt.

2.1 Generierung der Graubilder

Der erste Schritt der Bildvorverarbeitung ist die Generierung von Graubildern aus den Eingangswerten. Sie dienen der Vermeidung von unterschiedlichen Ergebnissen, die entstehen würden, wenn man die folgenden Schritte einzeln auf die Farbkanäle angewendet würde. Dafür werden die zwei Farbbilder der Funktion **rgb_to_gray.m** übergeben. Mit Hilfe der aus der Hausaufgabe bekannten Gleichung (1) werden die Grauwerte für jeden Pixel aus den Farbkanälen errechnet und auf ganzzahlige Werte gerundet.

$$gray_value = \text{uint8}(0.299 * R + 0.587 * G + 0.114 * B) \quad (1)$$

In Gleichung (1) stehen R, G und B für die Werte der Farbkanäle Rot, Grün und Blau.

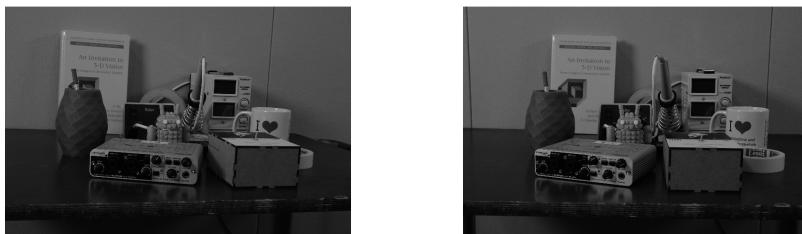


Abbildung 2: Generierte Graubilder GL und GR

2.2 Harris Detektor

Nun werden in den Graubildern Merkmale zu gesucht. Dafür wird der Harris-Eckendetektor aus der Vorlesung verwendet. Dieser unterscheidet drei Merkmale: eine homogene Fläche, eine Kante und eine Ecke. Um diese Unterscheidung zu erreichen, wird zunächst der Bildgradient mit einem vertikalen und horizontalen Sobel-Filter approximiert und aus dem Ergebnis die Matrix $G(x)$ aus Gleichung (2) erzeugt. Die Matrixeinträge entsprechen dabei den partiellen Ableitungen in x und y Richtung.

$$G(x) = \begin{pmatrix} G_{11} & G_{12} \\ G_{21} & G_{22} \end{pmatrix} \quad (2)$$

Mit Hilfe der Matrix $G(x)$ können nun durch Anwendung der Gleichung (3) die verschiedenen Merkmale extrahiert werden. Dabei gilt folgende Interpretation: Ist H größer als ein positiver Schwellwert τ , bzw. sind beide Eigenwerte groß, so handelt es sich um eine Ecke, den markantesten Merkmalstyp. Wird H kleiner als ein negativer Schwellwert $-\tau$, bzw. ist ein Eigenwert ist groß und der andere klein, so wird an dieser Stelle eine Kante detektiert. Für H -Werte die nahe null oder betragsmäßig klein sind, sind beide Eigenwerte klein. An diesen Stellen wurde eine homogene Fläche gefunden, die für die Erkennung von Korrespondenzen wertlos ist.

Die Parameter für den Harris-Eckendetektor sind in Tabelle 1 aufgeführt.

$$H = \det(G) - k(\operatorname{tr}(G)) \quad (3)$$

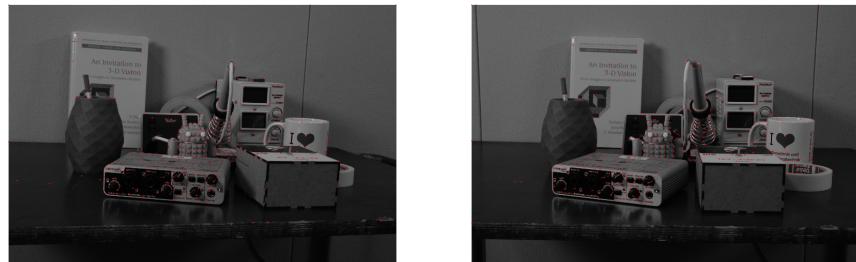


Abbildung 3: gefundene Merkmalspunkte des Harris-Eckendetektors

Parameter	Beschreibung	Standard-Wert	ermittelter Wert
Harris-Detektor			
segment_length	Größe des betrachteten Bildsegments	15	3
k	Gewichtung zwischen Ecken- und Kantenpriorität	0.05	0.02
tau	Schwellwert zur Detektion einer Ecke	10^6	10^{-4}
min_dist	minimaler Pixelabstand zweier Merkmale	20	30
tile_size	Kachelgröße	[200,200]	[20,30]
N	maximale Anzahl der Merkmale in einer Kachel	5	400
RanSaC			
epsilon	geschätzte Wahrscheinlichkeit der Wahl eines Ausreißers bei einem zufällig gewählten Paar	0.5	0.65
tolerance	Schwellwert zur Akzeptanz eines Paars	0.01	5
p	gewünschte Wahrscheinlichkeit eines Ergebnissatzes von korrespondierenden Punktpaaren ohne statistische Ausreißer	0.5	0.95

Tabelle 1: Parameter für Harris-Eckendetektor und RanSaC

Die große Diskrepanz zwischen den gewählten und den Standardparametern kommt daher, dass vor der Merkmalssuche die Kanten in den Bildern mit Hilfe eines Gaußfilters geglättet wurden. Dadurch müssen die Parameter angepasst werden.

2.3 Korrespondenzen

Die gefundenen Merkmale sollen in diesem Schritt zu Korrespondenzen zwischen den Bildern gepaart werden. Die Bildsegmente W werden mit der Gleichung eq:corres normalisiert und durch Normalized-Cross-Correlation, Gleichung (5), auf Ähnlichkeit geprüft.

Liegt der Wert nahe bei +1, so impliziert dies eine hohe Korrelation. Dieses Ergebnis lässt sich als Korrespondenz interpretieren.

$$W = \frac{1}{\Sigma(W)} (W - \bar{W}) \quad (4)$$

$$NCC = \frac{1}{N-1} \text{tr} (W_n^T V_n) \quad (5)$$

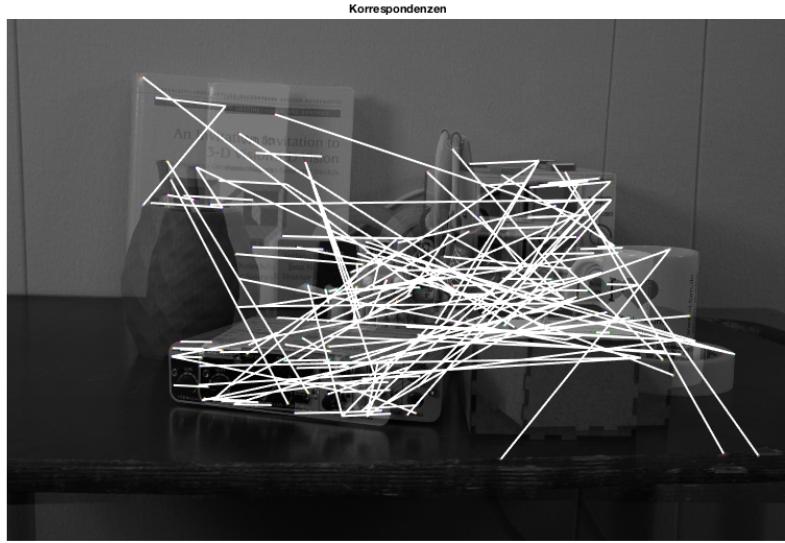


Abbildung 4: Korrespondenzen zwischen den beiden Bildern

2.4 RanSaC-Algorithmus

Um zu verhindern, dass Fehler beim Paaren der Korrespondenzen einen großen Einfluss auf die weiteren Schritte der Bildverarbeitung haben, werden die Paare durch ein RanSaC-Algorithmus gefiltert.

Das Ziel dieses Algorithmus ist es robuste Korrespondenzen zu finden. Dafür sind die folgenden Schritte notwendig:

1. Wahrscheinlichkeiten p (korrekte Lösung) und e (Ausreißer) festlegen
2. Datenmodell und Anzahl k an nötigen Parametern bestimmen
3. k zufällige Korrespondenzen wählen und die Parameter ermitteln
4. Menge M an Korrespondenzen bestimmen, die innerhalb der Toleranz liegen, mit Hilfe der Sampson-Distanz aus Gleichung (7)

5. Parameter und Menge M speichern, um die größte gefundene Menge zu ermitteln
6. Schritte 3-5 s-mal wiederholen, nach Gleichung (6), wiederholen
7. Finale Parameter aus der größten Menge M (Consensus-Set) bestimmen

$$s = \frac{\ln(1-p)}{\ln\left(1 - (1-e)^k\right)} \quad (6)$$

$$d_{sampson}\left(x_1^{(j)}, x_2^{(j)}\right) = \frac{\left(x_2^{(j)T} F x_1^{(j)}\right)^2}{|\hat{e}_3 F x_1^{(j)}|_2^2 + |x_2^{(j)T} F \hat{e}_3|_2^2} \quad (7)$$

Die Vorteile des RanSaC-Algorithmus liegen darin, dass ein Modell trotz Ausreißern bestimmt werden kann. Dabei kann der Algorithmus mit mehr als 50% Ausreißern umgehen. Als Nebenprodukt der Berechnung werden die Ausreißer identifiziert.

Der vergleichsweise hohe Rechenaufwand und die Tatsache, dass das Ergebnis nicht deterministisch ist, sind die Nachteile des RanSaC.

Die Abbildung 5 zeigt acht robuste Korrespondenzen, die nach dem RanSac-Algorithmus für die weitere Verarbeitung verwendet werden. Auf Grund der großen Unterschiede in den Parametern in Tabelle 1 wird nicht wie in der Literatur üblich der Parameter s aus Gleichung (6) als Abbruchkriterium im RanSaC-Algorithmus verwendet, sondern die Größe des Korrespondenzpaars. Diese muss für den weiteren Verlauf zwingend größer acht sein.



Abbildung 5: Korrespondierende Pixelpaare der beiden Bilder nach dem RanSaC-Algorithmus

2.5 Achtpunkタルゴリズム

Die durch den RanSaC-Algorithmus gefunden robusten Korrespondenzen werden nun zur Schätzung der Essentiellen Matrix verwendet. Für deren Findung kommt der sogenannte „Achtpunkタルゴリズム“ zum Einsatz.

Die Voraussetzung für diesen Algorithmus sind mindestens acht homogene Korrespondenzpunkpaare, die alle idealerweise die Epipolargleichung (8) erfüllen und linear unabhängig sind. Homogen bedeutet, dass die Punkte in der Form $P = [xy1]^T$ angegeben werden.

$$x_2^{jT} E x_1^j = 0 \quad (8)$$

Der erste Schritt ist die Bildung des Kronecker-Produkts der Koordinaten x_1 und x_2 nach Gleichung (9). Durch Vektorisierung der gesuchten essentiellen Matrix E ergibt sich das homogene lineare Gleichungssystem (10), das für $n > 8$ keine nicht-triviale Lösung besitzt. Zur Vermeidung dieses Problems wird das Gleichungssystem in das Minimierungsproblem aus Gleichung (11) umgeformt und dieses stattdessen gelöst. Da Lösungen skalierungs invariant sind, kann die Suche auf $x : \|x\|_2 = 1$ beschränkt werden. Die Singulärwertzerlegung von A löst dieses Problem mit dem Ergebnis (12).

$$A = x_1 \otimes x_2 = [x_1 x_2, x_1 y_2, x_1 z_2, y_1 x_2, y_1 y_2, y_1 z_2, z_1 x_2, z_1 y_2, z_1 z_2]^T \quad (9)$$

$$AE^s = 0 \quad (10)$$

$$G^s = \operatorname{argmin} \|AE^s\|_2^2 \quad (11)$$

$$G^s = v_9 \quad (12)$$

Im Anschluss wird G zurück in eine 3x3-Matrix gewandelt und durch eine weitere Singulärwertzerlegung auf die nächstgelegene essentielle Matrix im Vektorraum projiziert. Da E nur bis auf Skalierung geschätzt werden kann, werden in der Praxis die ersten beiden Singulärwerte gleich 1 gesetzt und der dritte zu null.

2.6 Rekonstruktion von Rotation und Translation

Das Ziel der Rekonstruktion ist es, aus der geschätzten essentiellen Matrix E die zugehörige Rotation und Translation zu berechnen.

Als erstes wird mittels Singulärwertzerlegung (13) von E die möglichen Kandidaten für R und T bestimmt. Dabei ist sicherzustellen, dass es sich bei $R1$ und $R2$ um Rotationsmatrizen handelt. Rotationsmatrizen erfüllen die Bedingungen (14) und lassen sich mit den Gleichungen (15) und (16) bestimmen.

$$[U, \Sigma, V] = \operatorname{svd}(E) \quad (13)$$

$$\det(R) = 1R^{-1} = R^T R_Z \left(\pm \frac{\pi}{2} \right) = \begin{pmatrix} 0 & 0 \mp 1 & 0 \\ \pm 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad (14)$$

$$R_{1,2} = UR_Z^T \left(\pm \frac{\pi}{2} \right) V^T \quad (15)$$

$$\hat{T}_{1,2} = UR_Z \left(\pm \frac{\pi}{2} \right) \Sigma U^T \quad (16)$$

Insgesamt werden vier mögliche euklidische Transformationen durch die Korrespondenzen erklärt, da R und T nur bis auf ihr Vorzeichen bestimmbar sind. Allerdings ist nur eine Transformation geometrisch plausibel.

Im nächsten Schritt wird die Tiefe der Raumpunkte geschätzt. Dies geschieht mit der Singulärwertzerlegung der Gleichung (17) und führt im idealen Fall zu den Tiefeninformationen λ .

$$\min \|Md\|_2^2 \quad (17)$$

Die Rekonstruktion der 3D-Koordinaten kann zum einen durch fehlerhafte Korrespondenzschätzungen und zum anderen durch Diskretisierungsfehler ungenau werden. Als Lösungsansatz besteht auf der einen Seite die Möglichkeit durch den RanSaC-Algorithmus robuste Korrespondenzen zu schätzen und auf der anderen Seite kann eine zusätzliche Schätzung von λ_2 oder die Wahl eines robusten

Triangulationsverfahrens, welches den Achtpunkタルgorithmus als Initialisierung verwendet, hilfreich sein.

3 Rektifizierung

Das Ziel dieser Funktion ist das Rektifizieren eines Bildpaars. Hier wird der Algorithmus von Fusielo et al. [5] verwendet. Rektifizieren bedeutet, dass alle optischen Achsen parallel ausgerichtet werden sowie, dass die Epipolarlinien kolinear miteinander sowie mit den Abtastlinien des Bildes werden. Dies hat den Vorteil, dass das Erstellen der Tiefenkarte sehr viel schneller und mit höherer Qualität durchgeführt werden kann, da nur noch Reihen während des Verfahrens abgesucht werden müssen. Der Algorithmus wird im folgenden näher erläutert. [9]

3.1 Rektifizierung der Kameramatrizen

Der erste Schritt dieses Algorithmus beinhaltet die Rotation der jeweiligen Perspektivischen Projektionsmatrizen (18) um die optischen Zentren der Bilder, bis die Brennebenen koplanar zueinander stehen. Dies richtet die Epipolarlinien parallel zueinander aus [5].

In Gleichung (18) bis (20) wird die Rotation mathematisch dargestellt. K bezeichnet hier die intrinsischen Kameraparameter, T die Translation, R die Rotation und P_o bzw. P_n die alte und neue Projektionsmatrizen. Mit Blick auf die neue Perspektivische Projektionsmatrix aus Gleichung (19) fällt auf, dass diese immer noch die gleichen intrinsischen Kameraparameter besitzt, sich jedoch die Rotation verändert hat. Das bedeutet, dass sich nur die optischen Zentren verändert haben und das System als eine Kamera betrachtet werden kann, die entlang der X-Achse bewegt wird.

Nun wird mit der Gleichung (19) die neue Rotation näher erläutert. Da eine Rotationsmatrix konstruiert werden soll, werden alle Zeilen orthogonal zueinander ausgerichtet und normalisiert. Zuerst wird x in R also r_1 parallel zur Grundlinie ausgerichtet, indem die optischen Zentren c_1 und c_2 voneinander abgezogen und - wie oben erwähnt - normalisiert werden. Der nächste Basisvektor r_2 wird mithilfe eines zufälligen Vektors k orthogonal zu r_1 konstruiert. Analog wird r_3 orthogonal zu r_2 und r_1 ausgerichtet. Wenn alle 3 Zeilen sowie K und das optische Zentrum kombiniert werden, entsteht die neue Perspektivische Projektionsmatrix P_n .

$$P_o = K * [R \quad T] \quad (18)$$

$$P_n = K * [R \quad -R * c] \quad (19)$$

$$R = \begin{bmatrix} r_1 = \frac{c_1 - c_2}{\|c_1 - c_2\|} \\ r_2 = k \wedge r_1 \\ r_3 = r_1 \wedge r_2 \end{bmatrix} \quad (20)$$

3.2 Transformation

Der letzte Schritt in diesem Algorithmus beinhaltet das Mapping der Bildebenen von der neuen und alten Perspektivischen Projektionsmatrix P_n und P_o . Wie in Gleichung (21) und (22) zu sehen, werden die vorderen Teile der Projektionsmatrizen verwendet und jeweils eine Translation berechnet. Angewendet auf die Bilder wird durch die Translation die rektifizierte Version der Bildpaare erstellt.

$$P = [Q \quad q] \quad (21)$$

$$T = Q_n * Q_o^{-1} \quad (22)$$

3.3 Ergebnisse

In diesem Teil der Dokumentation werden die Ergebnisse des Rektifizierungsalgorithmus präsentiert. Abbildung 6 ist deutlich mehr verschoben als die Bilder in Abbildung 7. Das liegt daran, dass das 2. Bildpaar von Natur aus gerader ist als das erste. Jedoch kann in beiden Fällen beobachtet werden, dass die vertikalen und horizontalen Linien gerade ausgerichtet wurden. Der Algorithmus wurde ausgewählt, da er schnell und effizient ist. Außerdem ist er vergleichsweise leicht zu verstehen.



Abbildung 6: Rektifiziertes Bildpaar 1



Abbildung 7: Rektifiziertes Bildpaar 2

4 Tiefeninformationskarte

Mit den generierten rektifizierten Bilder kann nun im Folgenden eine Tiefeninformationskarte erstellt werden. Die Motivation für die Berechnung der Tiefeninformationskarte ist, das mit ihrer Hilfe für jeden Bildpunkt ein passender Raumpunkt zu konstruiert werden kann. Dafür werden Korrespondenzen mit der Summe der absoluten Differenzen gesucht und über eine Kostenfunktion die Ungleichheit der Bilder extrahiert. Daraus resultiert eine Tiefekarte.

Im Folgenden werden zwei verschiedene Ansätze zur Berechnung der Tiefeninformationskarte vorgestellt: der *Region Based Stereo Matching* Algorithmus und das *Block-Matching mit SAD*. Im weiteren Verlauf beruhen die Ergebnisse auf dem *Block-Matching mit SAD*.

4.1 Region Based Stereo Matching Algorithmus

Der erste Ansatz ist der *Region Based Stereo Matching Algorithmus* von [2], um die Tiefeninformationen aus einem Stereofarbbildpaar zu extrahieren. Dazu wird *Line Growing Based Stereo Matching* verwendet, was auf *Region Based* basiert. Dieser Algorithmus wird in drei Schritte geteilt:

Schritt 1: Ein geeigneter Startpunkt muss gefunden werden, der sich in keiner wachsenden Region befindet. Dann wird die Disparität dieses Punktes mit Hilfe der Energiefunktion (23) mit einem Zeilenfenster ausgerechnet werden.

$$e(i, j, d) = \frac{1}{3 \cdot n \cdot m} \cdot \sum_{x=i}^{i+n} \sum_{y=j}^{j+m} \sum_{k=1}^3 (L(x, y + d, k) - R(x, y, k))^2, \quad (23)$$

wobei $L(i, j, c)$ das linke und $R(i, j, c)$ das rechte Bild im RGB-Format, $n \times m$ die Fenstergröße und d die Disparität ist.

Schritt 2: Nun muss die Fehlerenergie des benachbarten Punkts zum Startpunkt berechnet werden. Falls diese kleiner ist als der zuvor gesetzte Fehlerenergieschwellwert, wird dieser Punkt stattdessen mit der Region assoziiert. Andernfalls wird Schritt 1 wiederholt, um einen neuen Startpunkt zu finden.

Schritt 3: Die Schritte 1 und 2 müssen werden für alle Zeilen des Bilds wiederholt. Wachsende Disparitätsregionen bilden dann die Disparitätskarte $d(i, j)$. Jedoch haben wir uns gegen diesen Algorithmus entschieden, da die Ergebnisse ungenauer sind als die Tiefeninformationskarte von 4.2.

4.2 Block-Matching mit SAD

Der zweite Ansatz ist das sogenannte *Block-Matching mit SAD*. Dabei wird mit Hilfe von absoluten Differenzen und darauf beruhenden Kosten ein Tiefeninformationskarte berechnet.

4.2.1 Summe der absoluten Differenzen

Als Vorbereitung für die Summe der absoluten Differenzen - SAD, werden ein Fenster, siehe Abbildung 8, definiert in dem im Anschluss nach korrespondierenden Bildpunkten gesucht wird. Da mit rektifizierten Bildern gearbeitet wird, müssen alle Korrespondenzen horizontal liegen. Das bringt den Vorteil mit sich, dass nur zeilenweise gesucht werden muss, wie Abbildung 9 zeigt.

Die Bildung der SAD liefert eine positive Zahl. Sie entsteht durch die Bildung der Differenz der beiden Eingabebilder. Diese Zahl kann als Maß der Unterschiedlichkeit zwischen den Bildern interpretiert werden.

Die Berechnung erfolgt durch Subtraktion der Farbwerte der Bilder für jedes Pixel anhand der Gleichung (24). Im Anschluss daran, wird für jede Disparity eine Kostenfunktion mit der Gleichung (25) gebildet.

Ausgewertet wird durch Betrachtung der Werte, wobei kleine Kosten für Ähnlichkeit und richtige Disparity stehen.

$$cost_block = block_left - block_right \quad (24)$$

$$cost = \sum \sum |cost_block| \quad (25)$$



Abbildung 8: Fensterdefinition für SAD nach [6]



Abbildung 9: Korrespondenzsuche mit SAD nach [6]

Beruhend auf diesen Ergebnissen werden sogenannte Sub_Pixel mit der Gleichung (26) bestimmt und dienen der Verifikation der Tiefeninformation. Dabei stehen die Parameter C_2 für den SAD-Wert am nächsten passenden Block und C_1 bzw. C_3 für den Wert link und rechts von C_2 . d_2 ist die Ungleichheit am untersuchten Block.

Im Unterschied zur reinen Betrachtung der Kosten des jeweiligen Blocks werden hier die Werte der Nachbarblöcke mit einbezogen. Dies hat den Vorteil, präzise Aussagen über die verglichenen Blöcke zu treffen.

Im Anschluss wird die Disparitymap mit den geringsten Kosten ausgewählt. Diese ist in Abbildung 10 dargestellt.

$$d_{est} = d_2 - \frac{1}{2} \frac{C_3 - C_1}{C_1 - 2C_2 + C_3} \quad (26)$$

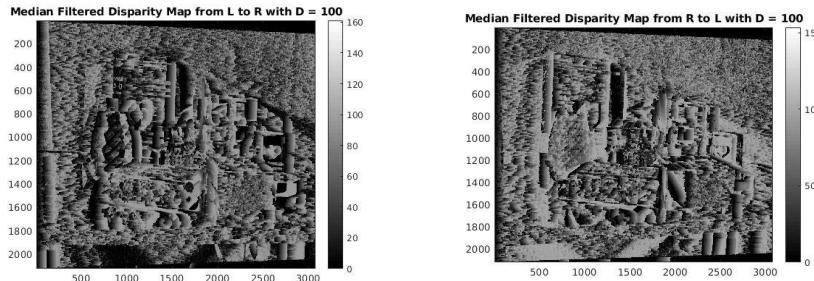


Abbildung 10: Disparitymaps

4.2.2 Erstellung der Tiefeninformationskarte

Die eigentliche Tiefeninformationskarte (Depthmap) wird durch die Gleichung (27) nach [8] berechnet.

Da die Parameter b (Grundlinie) und f (Brennweite) lediglich skalare Faktoren sind, ist die Tiefenkarte das Inverse der Disparitymap. Abbildung 11 zeigt das Ergebnis der Tiefeninformationsberechnung. Dabei kommt als finaler Abschluss ein Median-Filter nach [3] zum Einsatz. Dessen Aufgabe ist es, Störpixel im Bild auszugleichen, um eine robuste Karte zu erhalten. Dabei werden in einem definierten Fenster die Nachbarn des Pixels durch den Mittelwert der Grauwerte

ersetzt. Dabei findet keine Kantenglättung statt.

$$d = \frac{bf}{Z} \quad (27)$$

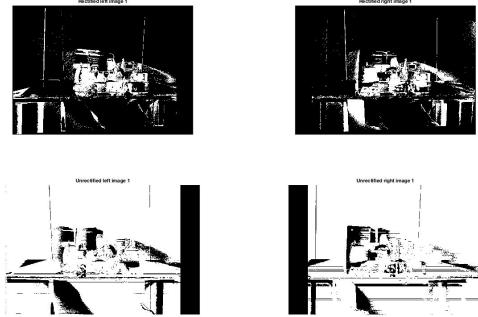


Abbildung 11: Depthmap

5 Rendering basierend auf Tiefeninformationen

Nachdem die Tiefenkarte anhand der rektifizierten Bildern erstellt wurde, kann nun ein Rendering-Algorithmus angewendet werden, um die virtuelle Ansicht zu generieren. Zwei Herangehensweise werden im Folgenden erläutert.

5.1 Rendering mit Inverse Mapping

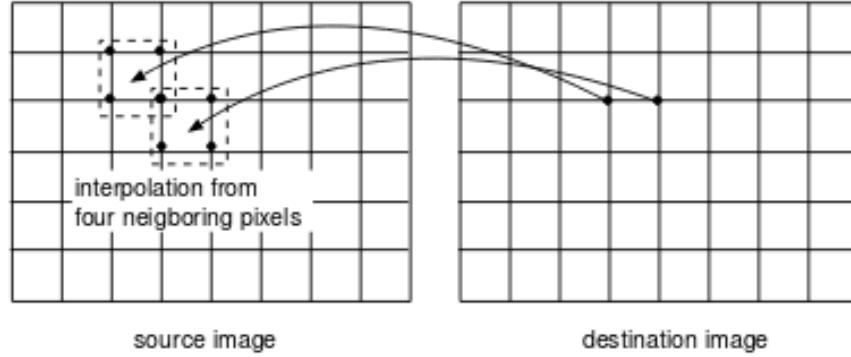
Für das Rendering wird die Inverse Mapping Theorie von Morvan [7] verwendet. Wie man in Abbildung 12 sehen kann, werden die Zielpixel auf das Gitter des Originalbild projiziert, wobei die Farbe des Zielpixels aus der Farbe der benachbarten Pixel aus dem Originalbild interpoliert wird. Ein Vorteil dieser Methode ist laut dem Autor, dass alle Pixel des Zielbildes richtig definiert werden und keine Löcher im generierten Bild auffindbar sind.

Der Inverse Mapping Algorithmus wird in 4 Schritte geteilt:

Schritt 1: Die Tiefenkarte des Originalbilds wird in die Position des Zielbildes verzerrt, was mit der 3D Image Warping Gleichung (28) erreicht wird. Hierbei sind \mathbf{d}_1 und \mathbf{d}_2 die Tiefenpixelkoordinaten des Ursprungs- bzw. Zielbildes und λ_2 ein positiver Skalierungsfaktor. Der Tiefenwert Z_{1w} ist durch den Pixelwert an der Koordinate \mathbf{d}_1 im Originalbild definiert. Außerdem korrespondieren $(\mathbf{K}_2, \mathbf{R}_2, \mathbf{C}_2)$ und (\mathbf{K}_1) zu den Kameraparametern des Ziel- bzw. Originalbilds. Falls Löcher in diesem Schritt auftauchen werden sie noch nicht aufgefüllt.

$$\lambda_2 \mathbf{d}_2 = \mathbf{K}_2 \mathbf{R}_2 \mathbf{K}_1^{-1} Z_{1w} \mathbf{d}_1 - \mathbf{K}_2 \mathbf{R}_2 \cdot \mathbf{C}_2 \quad (28)$$

Abbildung 12: Inverse Mapping von [7]



Schritt 2: Um undefinierte Pixel aus dem Forward Mapping Algorithmus zu vermeiden wird eine Dilationsoperation auf das gerenderte Tiefenbild angewendet. Als nächstes werden zwei Erosionsoperationen angewendet um die Abgrenzung der Pixel im Vordergrund zu reduzieren. So wird sicher gestellt, dass vermischtte Pixel an Objekträndern nicht als Pixel im Vordergrund klassifiziert werden. Bei der Implementierung wurden Filter zwar ausprobiert, jedoch weggelassen, da keine Parameter gefunden wurden, die das Bild verbessert hätten.

Schritt 3: Für jedes definierte Pixel \mathbf{d}_2 des Zieltiefenbilds wird ein korrespondierender 3D Weltpunkt $(X_{2w}, Y_{2w}, Z_{2w})^T$ durch Rückwärtsprojektion (29) ausgerechnet.

$$\begin{pmatrix} X_{2w} \\ Y_{2w} \\ Z_{2w} \end{pmatrix} = \mathbf{C}_2 + \lambda \mathbf{R}_2^{-1} \mathbf{K}_2^{-1} \mathbf{d}_2. \quad (29)$$

λ ist hier definiert als

$$\lambda = Z_{2w} - C_{2z} r_3 \text{ mit } \begin{pmatrix} r_1 \\ r_2 \\ r_3 \end{pmatrix} = \mathbf{R}_2^{-1} \mathbf{K}_2^{-1} \mathbf{d}_2 \text{ und } \mathbf{C}_2 = \begin{pmatrix} C_{2x} \\ C_{2y} \\ C_{2z} \end{pmatrix}, \quad (30)$$

mit den Tiefenwerten Z_{2w} , die durch die Pixelwerte bei der Koordinate \mathbf{d}_2 im Zieltiefenbild definiert sind.

Schritt 4: Der berechnete 3D Punkt wird dann auf die Originalbildstruktur projiziert mit

$$\lambda p_1 = \underbrace{[\mathbf{K}_1 | \mathbf{0}_3] \begin{bmatrix} \mathbf{R}_1 & -\mathbf{R}_1 \mathbf{C}_1 \\ \mathbf{0}_3^T & 1 \end{bmatrix}}_{\text{Projektionsmatrix}} \begin{pmatrix} X_{2w} \\ Y_{2w} \\ Z_{2w} \\ 1 \end{pmatrix}, \quad (31)$$

sodass die Farbe des Zielpixels p_2 , aus den Farbwerten der Pixeln, die p_1 im Originalbild umgeben, interpoliert werden kann.

Dieser Algorithmus wird auf beide Input-Bilder angewendet, einmal von der linken Seite und einmal von der rechten Seite. Dabei wird je nach dem Drehfaktor p das erste Bild mit $(1-p)$ und das zweite Bild mit p gewichtet und zum Schluss zu einem finalen Bild zusammengeführt.

5.2 Rendering mit 3D Warping

Dieser Rendering Algorithmus basiert hauptsächlich auf dem Prinzip des 3D Warpings. Erst werden Schattenkonturen entfernt, dann das 3D Warping angewendet und zuletzt die Tiefenkarten bzw. generierten Bilder zusammengeführt. Im folgenden werden alle Schritte erklärt.

5.2.1 Entfernen der Schattenkontur

Zuerst wird der Algorithmus zur Beseitigung der Schattenkonturen erläutert. Das Ziel dieses Teils ist es, falsch gesetzte Pixel, wie in Abbildung 13 mittig, welche in Farbbildern auftreten, zu eliminieren. Die Quelle dieser Pixel sind Ecken im Bild, die mit Pixeln aus dem Vordergrund kontaminiert wurden. [4] Der erste Schritt dieses Algorithmus ist die Anwendung eines klassischen Canny Filter auf die Tiefeninformationskarte zur Detektion von Kanten (siehe Abbildung 13 links). Die detektierten Pixel werden dilatiert und die Pixel innen werden zwischen Vorder- und Hintergrund aufgeteilt. Schließlich werden Pixel im Farbbild, welche von Kanten aus dem Hintergrund aus verzerrt wurden, durch das Filtern ihrer Nachbarn mit dem Medianfilter ersetzt. Das rechte Bild in Abbildung 13 zeigt das finale Ergebnis [4].

Abbildung 13: Beispiel für Schattenkonturen [4]



5.2.2 3D Warping

Nachdem störende Pixel durch das Entfernen der Schattenkonturen entfernt wurden wird nun das eigentliche Rendering anhand von 3D Verzerrung angewendet.

Das im folgende beschriebene Verfahren basiert darauf, Pixel der Referenzbilder horizontal zu verschieben. Die Weite der Verschiebung wird dabei von dem Wert der Pixel in der Tiefeninformationskarte skaliert [1].

Wie in Gleichung (32) und (33) zu sehen, werden die Originalbilder I_0 und I_1 , sowie die dazugehörigen Tiefeninformationskarten d_0 und d_1 anhand der Tiefeninformation an die Stelle (i, j) verschoben [1]. θ bestimmt den Punkt der virtuellen Ansicht, indem es die Tiefeninformation skaliert. Die Originalbilder werden einfach nur verschoben, bei den Tiefeninformationskarten hingegen werden an den verschobenen Pixeln aktuelle Tiefeninformationen eingesetzt: Die jeweiligen Brennweiten $f_{0/1}$ und die Grundlinie werden hier durch die jeweilige Tiefeninformation geteilt [1].

$$I_{0/1}^t(i, j - \theta * d_{0/1}(i, j)) = I_{0/1}(i, j) \quad (32)$$

$$D_{0/1}^t(i, j + (1 - \theta) \theta * d_{0/1}(i, j)) = f_{0/1} \frac{b}{d_{0/1}(i, j)} \quad (33)$$

Es kann vorkommen, das die verschobenen Indizes keine ganzzahligen Werte ergeben. Deswegen werden, um möglichst viele Lücken im Bild zu vermeiden, die Indizes auf- bzw. abgerundet und eingesetzt. Bei mehreren Pixeln an einer der neuen Stellen hingegen wird der Wert mit der niedrigsten Tiefeninformation genutzt. [4]

5.2.3 Fusionieren der Bilder

Da nun 2 verschobene Ansichten generiert wurden, muss daraus eine virtuelle Ansicht erzeugt werden, indem I_0^t und I_1^t fusioniert werden [4].

Pixel an Stelle x , welche in beiden Bildern sichtbar sind und einen Tiefeninformationsunterschied $|D_0^t(x) - D_1^t(x)|$ unter 0,00001 haben, werden nach den Gleichungen (34) und (35) berechnet [4]. Der Wert 0.00001 wurde empirisch ermittelt. Wie in den unten stehenden Gleichungen zu sehen werden die Farbwerte der Bilder durch θ skaliert, da je nach Position der Ansicht das linke bzw. das rechte Bild ähnlicher zur virtuellen Ansicht sind.

$$I^t(x) = (1 - \theta) I_0^t(x) + \theta I_1^t(x) \quad (34)$$

$$D^t(x) = (1 - \theta) D_0^t(x) + \theta D_1^t(x) \quad (35)$$

Falls der Tiefenunterschied größer als 0.00001 ist, wird wieder der Wert mit der kleinsten Tiefeninformation verwendet. [4]

6 Ergebnisse

6.1 Ergebnisse mit Inverse Mapping

Beim ersten Versuch wurde die Inverse Mapping Methode für das Rendering benutzt, wobei das Bild aus Abbildung 14 generiert wurde. Deutlich im Bild zu



Abbildung 14: Fusioniertes Bild mit Inverse Mapping

erkennen sind viele schwarze Pixel, die nicht sinnvoll auf die Ursprungsbilder gemappt werden konnten. Das kann man auf die ungeeignete bzw. ungenaue Tiefeninformationskarte zurückführen, weil viele Punkte außerhalb der Bildebene liegen und somit nicht auf dem richtigen Bildpunkt abgebildet werden. Mit einer besseren Tiefeninformationskarte würden bessere Ergebnisse erzielt werden.

6.2 Ergebnisse mit 3D-Warping

Der Ansatz des 3D-Warping liefert im Vergleich zum Inverse Mapping deutlich bessere Ergebnisse. Besonders am Grünen Legomännchen ist beim Vergleichen eine Rotation zu erkennen. Abbildung 16 und 15 zeigt diese Ergebnis. Wie leicht zu erkennen ist, weist das Bild an oberen Rand einen schwarzen Streifen auf. Dies liegt daran, dass die Grundlage der Rektifizierung, die Tiefeninformationskarte, auf den sehr schwierigen Eingabebildern nicht optimal berechnet werden kann. Der Grund für vereinzelten schwarzen Pixel ist ebenfalls hier zu finden.

Das führt zu folgender Erkenntnis:

Die Ergebnisse sind akzeptabel aber nicht perfekt. Der Grund dafür liegt zweifelsohne in der Tiefeninformationskarte die auf den Inputbildern basiert. Mögliche Optimierungen wären zum einen bessere Ergebnisse durch Akzeptanz einer deutlich höheren Laufzeit oder optimierte Eingabebilder.

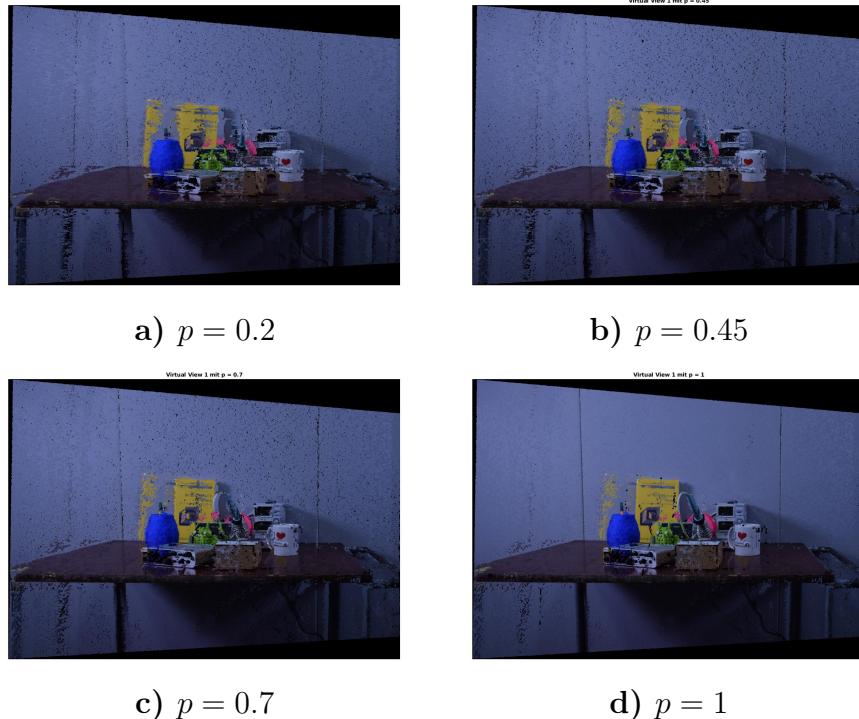


Abbildung 15: Fusionierte Bildpaare 1

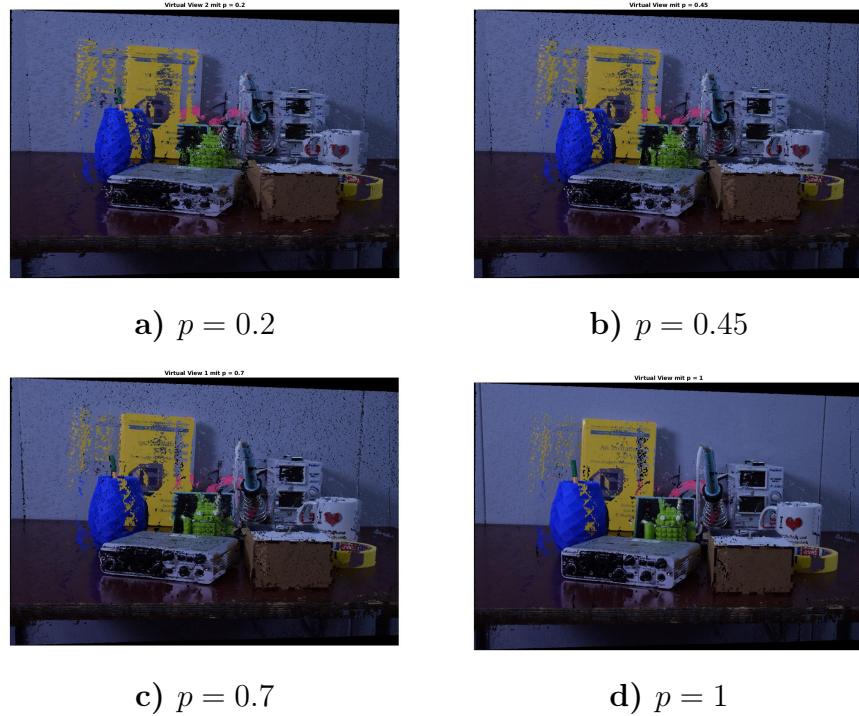


Abbildung 16: Fusionierte Bildpaare 2

7 Graphic User Interface

Als zusätzliches Feature wurde ein Graphical User Interface erstellt. Abbildung 17 zeigt die GUI beim ersten Start. Es besteht die Möglichkeit das Bildpaar L1/R1 oder das Paar L2/R2 zu laden. Des Weiteren kann ein beliebiger Blickwinkel p mit Hilfe einer Scrollbar unter den beiden Eingabebildern eingestellt werden. Der Standardwert liegt bei $p=0.5$.

Abbildung ?? zeigt die geladenen Bilder in der oberen Reihe. Durch einen Klick auf den *Generieren*-Button wird, wie in Abbildung 19 dargestellt, dass fertige Rekonstruktionsbild in der unteren Hälfte präsentiert. Zusätzlich wird unter dem Ergebnis die benötigte Zeit in Sekunden angezeigt.

Nach erfolgreicher Berechnung lässt sich das Programm durch erneutes wählen eines Bildpaars wieder starten.

Zur Ausführung muss **challengeGUI** im Command Window von Matlab eingegeben werden. Der aktuelle Fortschritt wird ebenfalls hier ausgegeben, um den Programmablauf nachvollziehen zu können.

Die aktuelle Version 2.4 dieser GUI, unterstützt nur das Laden und Berechnen der Bildpaare L1/R1 und L2/R2. Mit dieser Einschränkung verkürzt sich aller-

dings die Laufzeit der Anwendung erheblich, da die Tiefeninformationskarten im Voraus berechnet wurden und an der richtigen Stelle geladen werden.



Abbildung 17: Startbildschirm der GUI

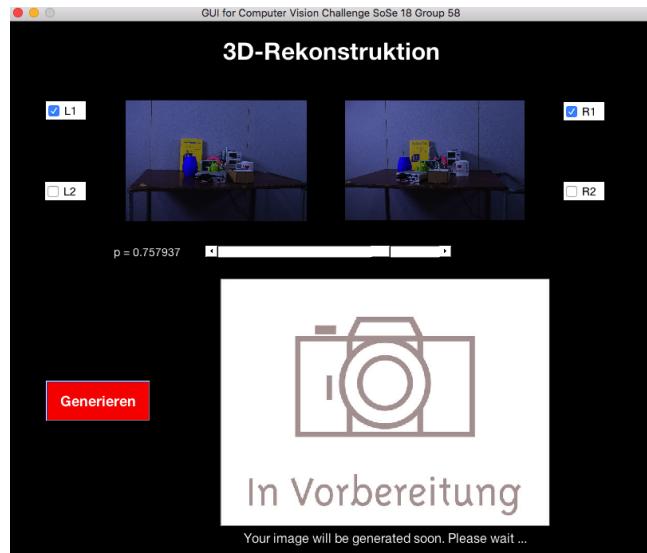


Abbildung 18: Anzeige für zwei geladene Bilder

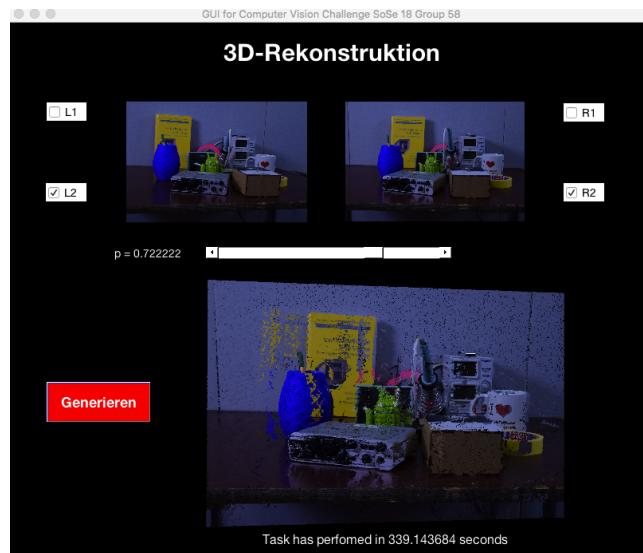


Abbildung 19: Darstellung mit rekonstruiertem Bild

Literatur

- [1] I. Ahn und C. Kim. „A Novel Depth-Based Virtual View Synthesis Method for Free Viewpoint Video“. In: *IEEE Transactions on Broadcasting* 59.4 (Dez. 2013), S. 614–626. ISSN: 0018-9316. doi: 10.1109/TBC.2013.2281658.
- [2] B. Baykant Alagoz. „Obtaining Depth Maps From Color Images By Region Based Stereo Matching Algorithms“. In: *OncuBilim Algorithm And systems Lab* 8.1 (2008).
- [3] Burge M.J. Burger W. *Digitale Bildverarbeitung*. 2015. ISBN: 978-3-642-04604-9.
- [4] J. Dai und T. Nguyen. „View synthesis with hierarchical clustering based occlusion filling“. In: *2017 IEEE International Conference on Image Processing (ICIP)*. Sep. 2017, S. 1387–1391. doi: 10.1109/ICIP.2017.8296509.
- [5] A. Fusiello, E. Trucco und A. Verri. „A Compact Algorithm for Rectification of Stereo Pairs“. In: *Machine Vision and Applications* 12.1 (2000), S. 16–22.
- [6] C. McCormick. *Stereo Vision Tutorial - Part I*. 2014. URL: <http://mccormickml.com/2014/01/10/stereo-vision-tutorial-part-i/>.
- [7] Y. Morvan. „Acquisition, compression and rendering of depth and texture for multi-view video“. In: *Eindhoven: Technische Universiteit Eindhoven DOI: 10.6100/IR641964* (2009), S. 93–98.
- [8] N. Navab. *Stereo Vision I: Rectification and Disparity*. 2011.
- [9] H. Syahputra u. a. „Improving disparity map of a specific object in a stereo image using camera calibration, Image rectification, and object segmentation“. In: 9 (Jan. 2014), S. 17939–17949.