

Computer Vision Challenge

Computer Vision

im Studiengang Master Elektro- und Informationstechnik

Sommersemester 2018

Gruppe 11:

Tim Janßen

Andre Thommessen

Oliver Doege

Julia Ströbel

Sebastian Hügler

Abgabe: 12.09.2018

Prüfer: [Prof. Dr.-Ing. Klaus Diepold](http://www.ldv.ei.tum.de/team/ordinarius/klaus-diepold/) / M.Sc. Stefan Röhrl

Inhaltsverzeichnis

[Einleitung 3](#_Toc523660887)

[Aufgabenstellung 3](#_Toc523660888)

[Idee und Umsetzung 5](#_Toc523660889)

[Umwandlung der Bilder in Graubilder 6](#_Toc523660890)

[Intensitäts- und Beleuchtungskorrektur 6](#_Toc523660891)

[Bilaterale Filterung 6](#_Toc523660892)

[Harris-Merkmale berechnen 6](#_Toc523660893)

[Korrespondenzschätzung 6](#_Toc523660894)

[Finde robuste Korrespondenzpunktpaare mit Hilfe des RANSAC-Algorithmus 6](#_Toc523660895)

[Berechne die Essentielle Matrix 6](#_Toc523660896)

[Bildrektifizierungsalgorithmus 6](#_Toc523660897)

[Disparitätsermittlung 7](#_Toc523660898)

[Ausgabe des Free-Viewpoint Bildes 7](#_Toc523660899)

[Verkürzung der Bearbeitungszeit 7](#_Toc523660900)

[Graphische Benutzeroberfläche (GUI) 8](#_Toc523660901)

[Abbildungsverzeichnis 10](#_Toc523660902)

[Literaturverzeichnis 11](#_Toc523660903)

# Einleitung

Die Computer Vision Challenge ist ein Bestandteil zur Bewertung der Leistungen der Studenten über ein Semester im Fach Computer Vision. Dieses Jahr besteht die Challenge darin, aus einem Stereo-Bild-Paar eine dritte virtuelle Ansicht zu generieren. Die Computer Vision Challenge ist eine Programmierarbeit, die in Gruppen bestehend aus drei bis fünf Personen durchgeführt wird. Es wird vom Lehrstuhl ein Skript vorgegeben, dessen Vorgaben zu erweitern sind. Einzelne Schritte sind zu kommentieren und Quellen auszuweisen.

Zusatzpunkte sind über eine optimierte Laufzeit und über eine GUI zu erreichen. Ein optimierter Code mit einer relativ kurzen Ausführungszeit wird positiv angerechnet. Eine graphische Benutzeroberfläche(GUI) wirkt sich ebenfalls positiv aus. Diese sollte zum Laden von Stereobildern, zur Einstellung des Blickwinkels der virtuellen Ansicht und zur Ausgabe der virtuellen Ansicht genutzt werden können.

# Aufgabenstellung

Die Challenge besteht darin, aus einem Stereo-Bild-Paar eine dritte virtuelle Ansicht zu generieren. Dabei soll der Blickwinkel der virtuellen Ansicht zwischen den beiden realen Ansichten liegen und durch einen Prozentwert frei bestimmbar sein. Das Programm soll in Matlab ohne die Hilfe spezialisierter Toolboxen erstellt werden. Als Grundlage liegen diverse Papers vor, die übers Semester verteilt wurden, dazu die Folien aus den Vorlesungen sowie die eigene Internetrecherche.

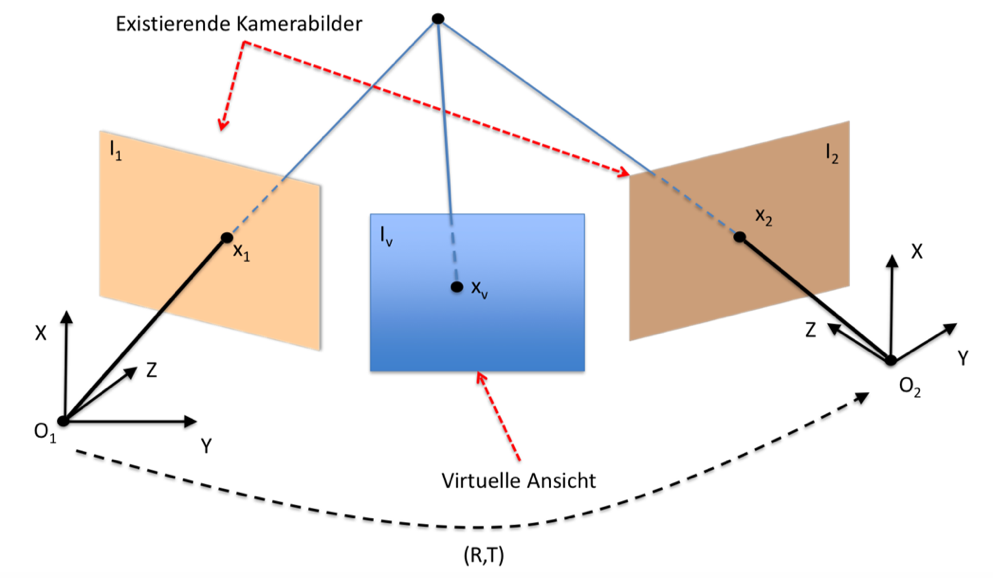


Abbildung 1: Geometrischer Zusammenhang zwischen den realen Bildern und einer künstlich generierten Ansicht

Abbildung 1 zeigt das zu untersuchende Szenario. Es werden zwei Bilder vom selben Motiv vorgegeben und mittels des zu erstellenden Programm übereinandergelegt, sodass ein drittes Bild berechnet wird.

Das vorgegebene Skript *challange.m* soll so erweitert werden, dass die Farbbilder *img/L1.jp*g und *img/R1.jpg* geladen und der Funktion *free\_viewpoint* übergeben werden. Die Funktion *free\_viewpoint* soll dann anhand des Parameters *p* eine virtuelle Ansicht zwischen den beiden realen Ansichten generieren.

Der Parameter *p* soll standardmäßig auf 50% stehen. Die Rechenzeit, die die Funktion *free\_viewpoint* benötigt hat, soll in der Variable *elapsed\_time* abgespeichert werden.



Abbildung 2: Linke Ansicht (L1.jpg)



Abbildung 3: Rechte Ansicht (R1.jpg)

# Idee und Umsetzung

In diesem Kapitel werden zunächst die Idee und der Ablauf des Programms skizziert. Danach werden die Aufgaben der einzelnen Programmbestandteile erläutert.

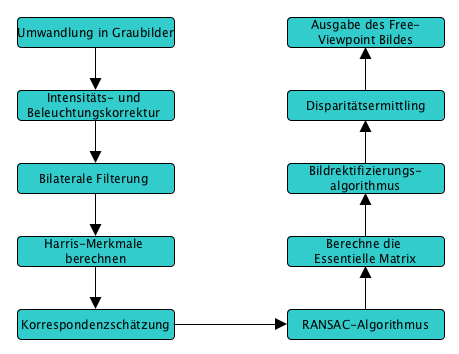


Abbildung 4: Ablauf des Programmes

Die Abbildung 4 zeigt den Aufbau und den Ablauf unseres Programms. Zuerst werden die beiden eingelesenen Bilder in Graubilder umgewandelt, um einen geringeren Aufwand bei den späteren Berechnungen zu erzielen. Danach wird eine Intensitäts- und Beleuchtungskorrektur bei beiden Bildern durchgeführt, um den vollen Intensitätsbereich ausnutzen und damit spätere Eckendetektionen zu erleichtern. Dieselbe Aufgabe verfolgt die darauffolgende bilaterale Filterung, bei der das Bild zwar weichgezeichnet wird, jedoch Kanten und Ecken erhalten bleiben. Im Anschluss erfolgt die Kanten- und Eckendetektion mit Hilfe des Harris-Detektors. Mit den detektierten Ecken aus den beiden Bildern kann eine Korrespondenzschätzung durchgeführt werden, um die Eckpunkte aus dem einen Bild den jeweiligen Eckpunkten aus dem anderen Bild zu zuordnen. Mit dem RANSAC-Algorithmus werden aus den geschätzten Korrespondenzen die robusten Korrespondenzpunktpaare bestimmt, um höhere Genauigkeit bei der Bestimmung der essentiellen Matrix zu erzielen. Diese erfolgt direkt im Anschluss mit Hilfe des Achtpunktalgorithmus. Durch die essentielle Matrix können die Rotation und Translation zwischen den beiden Ansichten berechnet werden. Diese werden zur Berechnung der dritten Ansicht benötigt. Denn mit der Rotation und Translation können die beiden Bilder rektifiziert werden, also eine Transformation durchgeführt werden, bei der die Epipolarlinien kollinear und parallel zur horizontalen Achse positioniert werden (Roth 2008). Somit wurden die Kamerabewegungen zwischen den beiden Bildaufnahmen ausgeglichen und die beiden Ansichten zeigen eine frontale Aufnahme nur um einen gewissen Abstand versetzt. Mit diesen rektifizierten Bildern können die Disparitäten bzw. eine Depth-Map berechnet werden. Aus dieser kann dann eine dritte rektifizierte Ansicht berechnet werden, welche im Anschluss wieder derektifiziert werden muss. Ansonsten passt die Ansicht nicht zu den beiden zu Beginn übergegebenen Bildern. Zuletzt wird das berechnete virtuelle Bild dem Benutzer ausgegeben.

Im Folgenden werden die hier kurz beschriebenen Schritte ausführlich beschrieben und erläutert.

## Umwandlung der Bilder in Graubilder

Nach der Übergabe der beiden Bilder, werden diese zunächst in Graubilder umgewandelt. Dies erlaubt in den späteren Schritten einfachere Berechnungen und somit auch einer kürzeren Berechnungszeit. Dazu wird die Funktion *rgb\_to\_gray* aufgerufen, die überprüft, ob es sich bei den beiden übergegebenen Bildern schon um Graubilder oder Farbbilder handelt. Bei Graubildern wird die Umwandlung übersprungen, bei Farbbildern werden die Intensitätswerte der drei Farben mit den in der Vorlesung vorgestellten Faktoren multipliziert und im Anschluss zusammenaddiert. Daraus entstehen dann die jeweiligen Intensitäten des berechneten Graubildes, welches von der Funktion zurückgegeben wird.

## Intensitäts- und Beleuchtungskorrektur

Die spätere Eckendetektion mit Hilfe des Harris-Algorithmus basiert auf lokale Intensitätsunterschiede innerhalb der Bilder. Um bei dem Algorithmus bessere Ergebnisse zu erzielen, soll der komplette Intensitätsbereich in den Bildern genutzt werden. Deshalb wird nach Graubildumwandlung die Funktion *gain\_offset\_correction\_cdf* aufgerufen, die die Skalierung der Intensitätswerte mit Hilfe der Histogramme der beiden Bilder vornimmt. Dazu werden die Histogramme der beiden Bilder ausgelesen und im Anschluss über den gesamten Intensitätsbereich gestreckt. Dadurch werden auch die einzelnen Intensitäten in den Bildern verändert bzw. auf den neuen skalierten Wert aus dem Histogramm gesetzt.

## Bilaterale Filterung

Um die Ergebnisse der anschließenden Eckendetektion mit dem Harris-Algorithmus weiter zu verbessern bzw. die Voraussetzungen zu optimieren, werden die beiden Bilder einer bilateralen Filterung unterzogen. Diese zeichnet die Bilder weich, wobei Kanten und Ecken erhalten bleiben, aber Rauschen reduziert wird. Somit treten diese noch mehr hervor und sind besser vom restlichen Bild zu erkennen. Dazu wird die Funktion *bfltGray* aufgerufen, welche die bilaterale Filterung auf einem durchführt. Für die Filterung wird ein Gaußfilter verwendet, der auf einzelne Bildfenster angewendet wird. Dabei wird der Kontrast innerhalb der einzelnen Bildfenster berücksichtigt, damit hohe Kontrastunterschiede wie Ecken und Kanten erhalten bleiben. Da die Filterung einen großen Teil der gesamten Programmlaufzeit einnimmt, in der dem Benutzer keine neuen Zwischenergebnisse angezeigt werden können, wird diesem ein Fortschrittsbalken angezeigt. Dieser zeigt dem Benutzer den prozentualen Fortschritt bei den Berechnungen an, so dass nicht der Anschein aufkommen kann, das Programm hätte sich eventuell auf gehangen.

Wie oben beschrieben, stellt die bilaterale Filterung einen großen Teil der Programmlaufzeit dar. Deshalb wird vor der Filterung die beiden Bilder durch die Funktion *downsample* auf eine geringere Auflösung bzw. Pixelanzahl heruntergerechnet und im Anschluss der Filterung durch die Funktion *upsample* wieder hochgerechnet. Durch die geringere Anzahl an Pixeln müssen weniger Berechnungen bei der bilateralen Filterung gemacht werden, weshalb die benötigte Berechnungszeit sinkt. In unserem Fall hat sich die Reduzierung der Auflösung um einen Viertel als optimal herausgestellt, wobei ein Mittelweg zwischen Berechnungsreduzierung und Qualitätsverlust bei den Ergebnissen betrachtet werden musste.

Das Herunterrechnen innerhalb der Funktion *downsample* erfolgt durch das Zusammenfassen von Pixeln und der Berechnungen eines neuen Intensitätswert, der die vorherigen Werte abbildet. Dabei liegt die Gewichtung auf den zentralen Pixeln innerhalb des jeweiligen betrachteten Fensters, welche durch einen Gaußfilter vorgenommen wird. Bei der Funktion *upsample* erfolgt das Gegenteil, es werden Zwischenwerte zwischen den einzelnen Intensitätswerten bestimmt. Dies erfolgt durch eine Interpolation.

## Harris-Merkmale berechnen

Nach den Vorbereitungen und Optimierungen für eine gute Eckendetektion innerhalb der Bilder, erfolgt diese durch den Harris-Detektor, der in der Funktion *harris\_detektor* implementiert wurde. Dieser wurde anhand der Vorlesung und den Hausaufgaben umgesetzt. Zunächst werden die Bildgradienten in x- und y-Richtung bestimmt, worauf eine Gewichtung der zentralen Pixel durch einen Gaußfilter und die Bestimmung der Harris-Matrix folgt. Mit dieser Matrix kann die Merkmalsdetektion durchgeführt werden, in dem nur Pixel als Merkmale klassifiziert werden, die nach der Harrismessung unter einem Schwellwert tau liegen. Danach erfolgt eine weitere Aussortierung bei den Merkmalen. Es werden nur Merkmale gespeichert, die innerhalb eines Fensters den größten Wert haben und somit die höchste Wahrscheinlichkeit für eine klare Ecke besitzen. Die anderen Merkmale innerhalb des jeweiligen betrachten Fensters werden nicht gespeichert. Im Anschluss gibt es eine Liste mit allen gefundenen Merkmalen in den beiden Bildern, welche in die Bilder eingezeichnet und dem Benutzer angezeigt werden.

## Korrespondenzschätzung

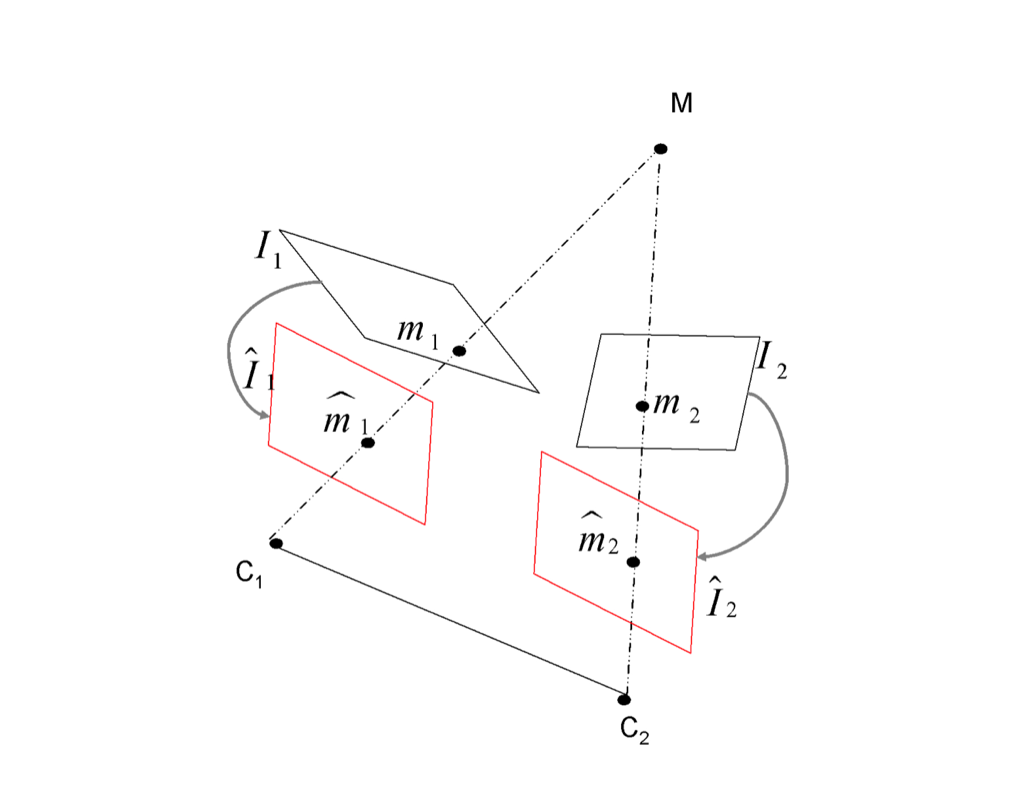
## Finde robuste Korrespondenzpunktpaare mit Hilfe des RANSAC-Algorithmus

## Berechne die Essentielle Matrix

## Bildrektifizierungsalgorithmus

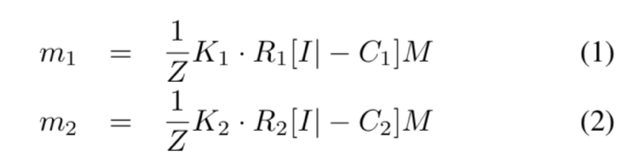
Ziel ist es die Epipole beider gegebenen Punkte gegen unendlich laufen zu lassen. Hier werden Bildpunkte der einzelnen Objekte jeweils der verschiedenen Bilder zugeordnet.

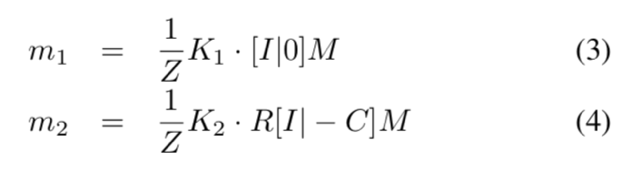
Bilder werden so gedreht, dass sie Parallel zueinander sind. (Paper Overview (erstes))

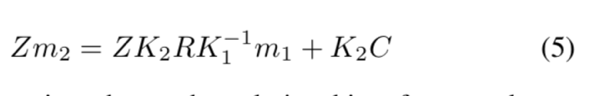


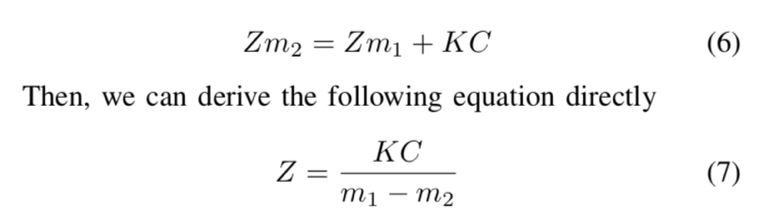
Parallele Epipolarlinien. 🡪 Suche auf den Linien nach gleichen Merkmalen (NNC-SAD)

Die Funktion sad\_scanline.m nnc\_scanline.m sorgt dafür, dass um ein Pixel ein gewisser Bereich (Window) gelegt wird, welches mit einem Bereich des anderem Bildes verglichen wird. Den Bereichen, die die geringste Abweichung aufweisen wird die gleiche Zugehörigkeit zugeordnet. Formel 1-7 Paper s.1025 , Berechnung Abstand, ... ,









Tiefenkarte Resultat, dadurch sind die Punkte im Raum bekannt und diese können auf die dritte, neue Fläche projiziert werden.

[bfltGray.m](https://github.com/juliatiptop/Gruppe-11/blob/master/Code/bfltGray.m)

[depth\_estimation.m](https://github.com/juliatiptop/Gruppe-11/blob/master/Code/depth_estimation.m)

[downsample.m](https://github.com/juliatiptop/Gruppe-11/blob/master/Code/downsample.m)

[get\_min\_max\_disparity.m](https://github.com/juliatiptop/Gruppe-11/blob/master/Code/get_min_max_disparity.m)

[ncc\_scanline.m](https://github.com/juliatiptop/Gruppe-11/blob/master/Code/ncc_scanline.m)

[sad\_scanline.m](https://github.com/juliatiptop/Gruppe-11/blob/master/Code/sad_scanline.m)

[upsample.m](https://github.com/juliatiptop/Gruppe-11/blob/master/Code/upsample.m)

## Disparitätsermittlung

Hier wird die Funktion get\_min\_max\_disparity aufgerufen.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | %GET\_LINE\_SEARCH\_INTERVALL Berechnet grob minimale und maximale Disparity | |  | % um den Liniensuchbereich zurück zu geben. | |  |  | |  | % Umwandlung in homogene Pixelkoordinaten | |  | anz\_kor = size(Korrespondenzen,2); | |  | Kor\_L = [Korrespondenzen(1:2,:);ones(1,anz\_kor)]; | |  | Kor\_R = [Korrespondenzen(3:4,:);ones(1,anz\_kor)]; | |  |  | |  | % Wende die Rectification Transformation auf die Korrespondenzpunktpaare an | |  | NEW\_Kor\_L = T\_rec\_1\*Kor\_L; | |  | NEW\_Kor\_R = T\_rec\_2\*Kor\_R; | |  |  | |  | % Bestimme disparities: d = x\_rec\_L - x\_rec\_R | |  | % Korrespondenzpunkt des linken Bildes liegt jeweils weiter rechts als der | |  | % des rechten Bildes | |  | disparities = NEW\_Kor\_L(1,:) - NEW\_Kor\_R(1,:); | |  | min\_disparity = min(disparities); | |  | max\_disparity = max(disparities); | |
|  |  |
|  | depth\_map = ... |
|  | depth\_estimation(img1\_rectified,img2\_rectified,K,T,offset\_x\_pixel,... |
|  | d\_cut\_up,d\_cut\_down,min\_disparity,max\_disparity); |
|  | %% Projektion auf neues Bild |
|  | disp('---------projection-----------') |
|  | alpha = p\*get\_max\_alpha(R); |
|  | for i=1:size(image1,3) |
|  | [virtual\_view\_img(:,:,i),img\_rectified\_new(:,:,i)] = projection(image1(:,:,i),... |
|  | image1\_rgb\_rectified\_full(:,:,i),... |
|  | depth\_map,... |
|  | K,-p,R\_rect,alpha,... |
|  | offset\_x\_pixel, d\_cut\_up,d\_cut\_down); |
|  | end |
|  | save('res\_28\_08\_V3\_try3') |
|  |  |
|  | figure |
|  | imshow(img\_rectified\_new); |

Vorstufe der Tiefenkarte

## Ausgabe des Free-Viewpoint Bildes

Die Bildgröße des dritten, berechneten Bildes wird auf die Bildgröße der gegebenen Bilder angepasst.

# Graphische Benutzeroberfläche (GUI)

Die graphische Benutzeroberfläche dient dem Bediener zur Steuerung des Programmes. Für ein schnelles zurechtfinden befinden sich auf der Oberfläche nur die wichtigsten Operatoren.



Abbildung 5: Benutzeroberfläche vor der Berechnung

Nach einer kurzen Erklärung, was dieses Programm leistet und wie es zu bedienen ist, kann der Blickwinkel, Parameter *p*, von 0 bis 1 eingestellt werden. Die richtige Eingabe wird durch eine minimale und maximale Grenze in dem Eingabefeld sowie einem Hinweis hinter dem Feld gewährleistet. Die gewünschte linke und rechte Ansicht wird jeweils über die Buttons *Linke Ansicht laden* und *Rechte Ansicht laden* ausgewählt. Dazu wird eine Explorer-Ansicht geöffnet, um die Bildauswahl für den Benutzer zu vereinfachen. Nach der Auswahl werden die Bilder in das Programm geladen und dem Benutzer zur Bestätigung jeweils unter den Buttons angezeigt. Zusätzlich ist der Pfad der Bilder direkt darunter einsehbar, damit der Benutzer kontrollieren kann, ob die richtigen Bilder geladen wurden und er sich nicht verklickt hat. Das dritte Bild kann dann über den Button *3. Ansicht berechnen* über das dahinter entworfene Programm berechnet werden. Nach der Berechnung wird das erzeugte Bild zwischen beiden zuvor geladenen Bildern angezeigt, um einen guten Vergleich zu den äußeren Ansichten zu geben. Die Berechnungszeit, die das Programm für die Berechnung der mittleren Ansicht benötigt, wird unterhalb des Bildes in Minuten angezeigt.

Die GUI wurde mit Hilfe des von Matlab zur Verfügung gestellten *App Designer* erstellt. Dieser bietet ebenfalls ein graphisches Interface im Sinne von *what you see, is what you get*. Dies erleichtert das Positionieren der einzelnen Elemente in dem Fenster. Zudem erzeugt es für die einzelnen Elemente den entsprechenden Code für deren Erzeugung. Danach konnten die Funktionen der Button und Eingabeoberflächen an den entsprechenden Stellen im Code implementiert werden. Das Einarbeiten in den App Designer erfolgte über die bereitgestellte Hilfe von Matlab.

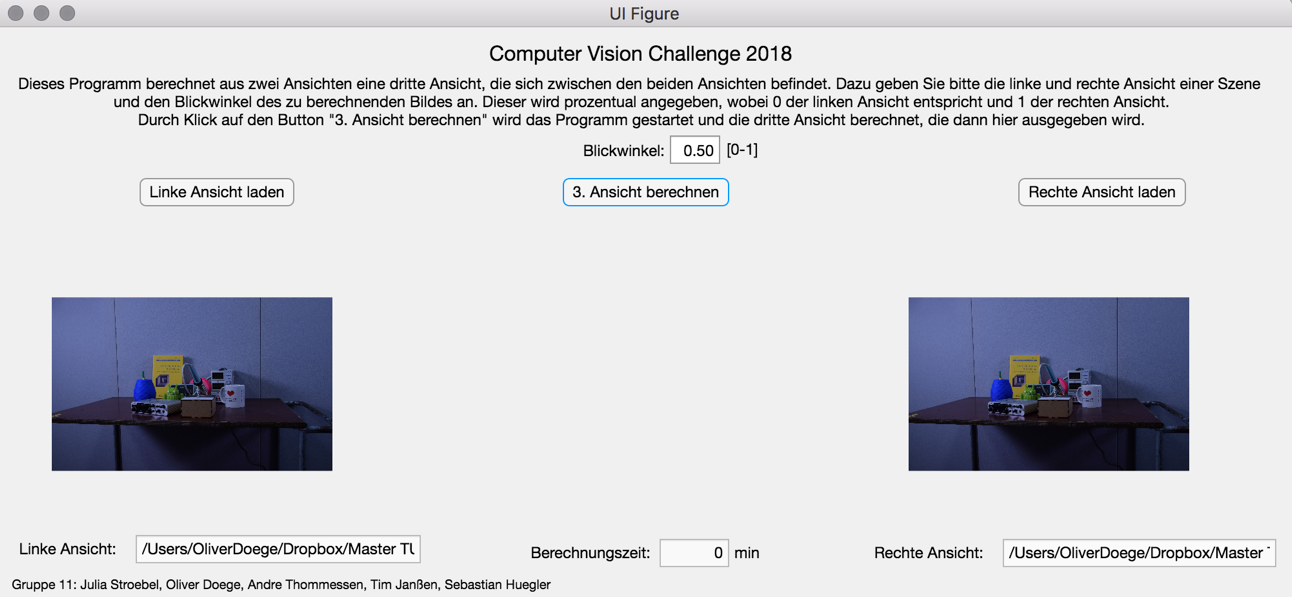


Abbildung 6: Benutzeroberfläche mit geladenen Bildern

# Abbildungsverzeichnis

[Abbildung 1: Geometrischer Zusammenhang zwischen den realen Bildern und einer künstlich generierten Ansicht 3](file://localhost/Users/OliverDoege/Desktop/Doku_Gr11_Challenge.docx#_Toc522452448)

[Abbildung 2: img/L1.jpg 4](file://localhost/Users/OliverDoege/Desktop/Doku_Gr11_Challenge.docx#_Toc522452449)

[Abbildung 2: img/R1.jpg 4](file://localhost/Users/OliverDoege/Desktop/Doku_Gr11_Challenge.docx#_Toc522452450)

[Abbildung 4: Projektverlauf 5](file://localhost/Users/OliverDoege/Desktop/Doku_Gr11_Challenge.docx#_Toc522452451)

# Literaturverzeichnis

Roth, Dr. Gerhard. 2008. *Rectification*. Zugriff: 02. September 2018.  
http://people.scs.carleton.ca/~c\_shu/Courses/comp4900d/notes/rectification.pdf

**1.1 Mindestanforderungen**

* **Dokumentation:** Erstellen Sie ein Dokument (\*.pdf, \*.docx), indem Sie die Funkti- onsweise ihres Programms kurz erklären und legen Sie auch die Quellen dar, die sie für Ihren Ansatz konsultiert haben. Nutzen Sie dazu gerne auch mathematische Be- schreibungen, Skizzen und Blockdiagramme. Fügen Sie in dieses Dokument auch die geforderten Werte, die Sie errechnet haben, sowie Plots, die Sie erstellt haben ein, und legen Sie das Dokument Ihrem Abgabearchiv bei. Weisen Sie in diesem Dokument auch auf eventuelle Zusatzfeatures hin, die Sie implementiert haben (sie- he Zusatzpunkte).

