CV-Challenge

Zeen Song, Shengzhi Wang, Xiajun Zhou, Lei Wang, Yikai Xu

July 2019

1 Introduction

In diesem Report haben wir die Ziel von 3D-Rekonstruktion mit der Hilfe von linkem und rechtem Kamera Bild erledigt. Der wichtigste Schritt für unseren Zweck ist eigentlich das Erhalten der Disparitäten. In Section 2 haben wir alle Algorithmus erklärt, die in unserer Arbeit für das Erhalten der Disparitäten inklusive sind. Section 3 zeigt wie wir die Aufgabe von "Verification", "Unittests" und "GUI" geschafft haben. In Section 4 erklärt es die Disparität Maps Ergebnisse von allen Methoden und auch eine Motorrad 3D-Rekonstruktion für die beste Methode. Zum Schluss haben wir in Section 5 zusammenfasst, wie die Ergebnisse aussehen und was wir gefunden haben in dieser Aufgabe. Und wir haben auch einen Appendix erstellt, in dem alle Ergebnisse der Verwendungen mit unterschiedlichen Methode auf verschiedene andere Bilder und die 3D-Rekonstruktionen mit der besten Methode für die anderen Bilder geplottet werden.

2 Algorithmus

Der Schwerpunkt dieser Aufgabe ist auffällig die Disparity Mapping. In diesem Abschnitt werden alle Kernel-Algorithmus für das Erhalten der Disparitäten erklärt, die in unserer Arbeit genutzt werden.

2.1 Block-Match

Block-Match (BM) Algorithmus ist ein Basis von den Stereo Matching Algorithmus. Die Ziel vom BM ist zum Vergleich der Bilder von der linken und rechten Kamera in Form einer kleinen verschiebenden Fenster, so dass die Disparitäten davon berechnet werden können. Eigentlich gibt es zwei Schiebefenster auf jeweils linke und rechte Bilder. Durch die Vergleichung jeden Pixels in ihre Fenster mit den entsprechenden Pixels in der anderen Fenster kann man die Disparität des zentralen Punktes der Fenster in dieser Position auf dem Bild bestimmt werden. Für die Vergleichung der Pixels in Schiebefenster gibt es eigentlich zwei folgende Methode, welche unsere Arbeit enthält:

Sum of Absolute Differences (SAD) SAD ist das Verfahren des lokalen Matching-Algorithmus. Lokalen Algorithm vergleichen einzelne Pixel oder lokale Ausschnitte aus einer Ansicht. Zur Korrespondenzfindung wird zu jedem Pixel eine Matching Kost zu anderen Pixeln berechnet. Je geringer diese Matching Kost, desto größer die Ahnlichkeit.

Die drei Hauptschritte sind:

1. Vorverarbeitung der Bilder durch rgb_to_gray function
2. Korrespondenzstuche entlang der horizontalen Epipolarlinien mit einem SAD-Fenster
3. Herausfiltern von schlechten Korrespondenzergebnissen

Besonders als zweiters werden korrespondierende Punkte zwischen den beiden Bildern gesucht. Dabei wird ein Suchfenster bestimmt. Nimmt es an, dass das Bild von linker Kamera Bild 1 und von rechter Kamera Bild 2 ist. Nach dem Maß auf Bild 1 durch ein Operator um eine adaptive Window_Size(ein quadratischer Block) zu bestimmen. Die Länge auf das Schiebenfenster wird durch "ndisp" in unserer Code gerechnet. Eine Suchfenster entlange der horizontalen Epipolarlinie geschoben und die am besten Überstimmung bestimmt. Während Jedes Umlaufs wird im Bild 1 von der linken Kamera an einer bestimmten Position das Suchfenster berechnet und im rechten Bild 2 an der Länge des Schiebenfenster Position links davon. Um die Ähnlichkeit zweier Block zu bestimmen, ergibt sich die SAD (Sum of Absolute Differences) (siehe Gleichung (1)) Werte nach flogender Formel:

$$\sum_{x,y \in W} |I_L(x,y) - I_R(x,y)| \tag{1}$$

Je niedriger der SAD-Wert von der zwei Block ist, desto ähnlicher sind sie. Die Verschiebung des am besten übereinstimmenden Blockes in Bild 2 im Vergleich zur ursprünglichen Position in Bild 1 ergibt sich die so genannte Disparität. Es kombiniert sich Disparität von allen Blöcke, ergibt es sich Disparität Map.

Census Transform Census Transform [1, 2, 3] ist auch eine Methode, die mit Hilfe der Schiebefenster auf zwei Bilder ist, damit die Ähnlichkeit zwichen beiden Bildern ausfinden werden kann. Im Vergeich zu SAD, die Census Transform Methode markiert erstens alle Pixel Werte als '1', welche größer als die zentralen Pixel Werte in den beiden Schiebefenster sind. Gleichzeitig werden die übrigen Pixel Werte auf '0' gesetzt. Nach der Sortierung diesen binären Werten wird es zwei Census Sequenzen erhalten. Die Korrelation zwishcen den beiden Schiebefenster wird durch die Berechnung der Hamming-Entfernung von den zwei Census Sequenzen erkennen. Je kleiner die Hamming-Entfernung, desto ähnlicher sind die beide zentralen Pixels von den beiden Schiebefenster.

Bei unserem Fall ist eine Kost Funktion verwendet, um die Korrespondenzpixel von einem Bild zu dem anderen Bild zu finden. Die optimalen Disparitäten sind erhalten durch die Minimierung der Kost Funktion. Und die Kost Funktion ist:

$$C(\mathbf{p}, d) = \rho(C_{census}(\mathbf{p}, d), \lambda_{census}) + \rho(C_{AD}(\mathbf{p}, d), \lambda_{AD}), \tag{2}$$

wo die $\mathbf{p}$ die Pixel Position ist, d die Disparity Niveau ist, C_{census} und C_{AD} die Hamming-Entfernung und die durchschnittliche Intensitätsdifferenz in RGB-Kanäle, ρ eine robuste Funktion auf die Kost. Folgende sind die Berechnungen von $\mathbf{p}$, C_{AD} und ρ:

$$\mathbf{p} = (x, y), \tag{3}$$

$$C_{AD}(\mathbf{p}, d) = \frac{1}{3} \sum_{i=R,G,B} \left| I_i^{Left}(\mathbf{p}) - I_i^{Right}(\mathbf{p} - (d, 0)) \right|, \tag{4}$$

$$\rho(c, \lambda) = 1 - exp(-\frac{c}{\lambda}) \tag{5}$$

In unseren Codes haben wir λ_{census} auf 1000 und λ_{AD} auf 10 gestellt. Die Cencus-Methode bewahrt die Positionseigenschaften der Pixel im Fenster und ist robust gegenüber Luminanzabweichungen. Einfach ausgedrückt, kann sie die durch den Beleuchtungsunterschied verursachte Fehlanpassung verringern.

Nach jeweils der oben gezeigten zwei Methoden bekommt man zwei Disparitäten: Eine von linkem Bild und eine von rechtem Bild. Nach dem Erhalt der Disparitäten muss noch eine Nachbearbeitung durchgeführt werden, sodass die Disparitäten glatter aussehen. Für diese Ziel ist eine Verfeinerungsmethode verwendet, welche erstens die sogenannte *Outlier-Detection* durchgeführt werden muss. Folgende ist die Kerngleichung für die Ausrechnung der Outlier-Detection auf dem linken Bild:

$$D_L(\mathbf{p}) = D_R(\mathbf{p} - (D_L(\mathbf{p}), 0)), \tag{6}$$

Es is auffällig dass die $\mathbf{p} = (x, y)$. $D_L(\mathbf{p})$ bedeutet die Disparität aus dem linken Bild für Pixel Position $\mathbf{p}$. Und D_R hat die gleiche Bedeutung für das rechte Bild. Für die Outlier-Detection der Disparität von rechtem Bild wird durch die ähnliche Gleichung wie (6) gerechnet. Seit die neuen Disparitäten von links und rechts bekommen, subtrahieren die beiden neuen Disparitäten mit den entsprechenden alten Disparitäten und ergeben sich zwei Fehleranpassung Matrizen. Jedes Element repräsentieret den Fehler der Disparität. Je größer das Element, desto ungenauer ist die Bestimmung der Disparität in dieser Pixel Position. Das heißt, es gibt höhere Möglichekeit, dieser Punkte eine Okklusion oder Fehleranpassung zu sein. Für diese Punkte haben wir auch zwei Methode verwendet: **Mean Filtering** und **Median Filtering**. Mean Filtering heißt es, wenn das Element von der Fehleranpassung Matrize größer als ein gewisser Wert (in unserer Arbeit haben wir auf 10 gestellt), werden eine Menge Nachbarn von der an den gleichen Pixel Position liegenden Pixel von den alten Disparität (in

unserer Arbeit haben wir diesen Punkte als zentraler Punkte und eine 20*20 Matrize erstellt) ausgewählt und rechnt den durschnittlichen Wert davon. Dieser Wert wird als neuer Wert den fehlerhaften Punkte ersetzen. Für die Median Filtering wählen die zentralen Punkten auch eine Menge Nachbarn. Der Unterschied ist, dass bei der Median Filtering der Medianwert als neuer Wert den fehlerhaften Punkte ersetzen. So ist eigentlich die zwei Verfeinerung-Prozesse.

2.2 Global-Match

Außer Block-Match Algorithmus haben wir noch ein Global-Match (GM) Algorithmus auch implementiert. Diese Idee ist inspiriert in Prinzip aus Block-Match. Im Block-Match wird ein kleines Fenster des rechten Bilds (normalerweise 8*8 Größe) erstellt, und wir vergleichen dieses Fenster mit dem Fenster mit gleicher Größe im linken Bild. Wenn der die Pixel-Differenz von beiden Fenster minimal ist, wird die Verschiebung des Fensters als Disparität markiert. Ähnlich wie in BM, wird in GM der ganzen rechten Bild mit dem ganzen linken Bild in pixelweise verglichen und verschoben. Während der Verschiebung wird auch eine Kostmatrize gegeben, die ständig die Pixel-Differenz für jedes Pixel im Bild erfasst. Diese Methode ist inspiriert von [4]. Die globale Kostmatrize (oder auch Energie Funktion genannt) wird duch folgende Gleichung angegeben:

$$E(d) = E_{data}(d) + \lambda E_{smooth}(d), \tag{7}$$

wo die Gleichung für $E_{data}(d)$ und $E_{smooth}(d)$ können wie folgende ausgerechnet:

$$E_{data}(d) = \sum_{(x,y)} C(x, y, d(x, y)), \tag{8}$$

$$E_{smooth}(d) = \sum_{(x,y)} \rho(d(x, y) - d(x + 1, y)) + \rho(d(x, y) - d(x, y + 1)), \tag{9}$$

wo C die Pixel Differenz zwischen dem linken und rechten Bild ist und ρ eine Identitätsfunktion ist. Für die Berechnung der obigen drei Gleichungen, wird diese Kostmatrize in jedem Kreislauf mit einer Mini-Kostmatrize verglichen, Die Koordinaten, deren Werte in der Kostmatrize kleiner als Mini-Kostmatrize sind, werden die kleinere Werte in Mini-Kostmatrize aktualisiert. Ähnlich wir in BM wird auch die Verschiebungsgröße als Disparität aufgeschrieben, wo die Mini-Kostmatrize am kleinsten sind. Die Anzahl der Kreisläufe wird durch die Größe der Parameter ndisp bestimmt. Aber beim zu groß ndisp muss der Wert von ndisp verkleinert wird. Der Grund dafür ist, dass wir im Programm immer nullmatrize hinzugefügt haben, um die Subtraktion zwischen zwei Bilder erfolgreich zu führen. Aber wenn das rechte Bild sich zu viel nach recht verschiebt, wird die Kostmatrize nicht mehr Pixel-Differenz, sondern Unterschied zwischen null und Bildpixel. In diesem Fall die minimalen Werte in Mini-Kostmatrize steht nicht für richtige Disparität. Im Programm wird ndisp mit 0,3 multipliziert,

wenn ndisp gleich wie die Breite des Bilds.

Die Nachbearbeitung ist auch notwendig im diesen Fall. Erstens ist die Verfeinerungsmethode sowie dargestellt in section 2.1 durchgeführt. Das Ergebnis von dieser Methode kann gesehen werden in section 4.2 von Abbildung 7(a). Allerdings funktionierte die Mean Filtering Verfeinerungsmethode nicht gut. Deswegen haben wir noch ein paar Filter verwendet, wie zum Beispile Gaussian Filter, um die Verfeinerung zu realisieren. Die genauen Ergebnisse der Verwendung unterschiedlicher Filter werden in Figure 7 dargestellt.

2.3 Beschleunigung der Prozessgeschwindigkeit

Die Prozessgeschwindigkeit spielt eine wichtige Rolle in der realen Anwendung. Um jeder Algorithmus beschleunigen zu können, haben wir verschiedene Methode übergelegt und ausprobiert. Folgende sind die unterschiedlichen Methode:

- **BM + SAD** Jedes Mal wenn wir die optimalen Disparitäten für die zentralen Pixel Punkten der Schiebefenster bekommen, aktualisieren wir nicht nur die zentralen Pixel Punkten, sondern auch alle Nachbarn in den gleichen Schiebefenster mit den gleichen Disparitäten Werte. Mit diesem Trick haben wir die Prozesszeit verbessert. Die Suchanzahl hat sich um Quadrat der Schiebefenstergröße reduziert. Es dauert ungefähr 4s für das Bild "playground".

- **BM + Census Transform** Der Trick in diesem Fall ist ähnlich wie der bei BM+SAD. Alle Nachbarn in den gleichen Schiebefenster werden mit den gleichen Disparitäten Wert von den zentralen Pixel Punkten aktualisiert. Der Unterschied liegt dran, dass bei BM + Census Transform wir die Bilder zum Erst das Down-Sampling Prozess durchführen. Bei diesem Fall ist die Suchanzahl um $(x^2 \cdot Win^2_{Size})$ reduziert, wo x die Anzahl des Verkleinerungsmehrfaches von Down-Sampling und Win_{Size} die Schiebefenstergröße sind. Es dauert ungefähr 50s für das Bild "playground".

3 Implementierung

3.1 GUI

Figure 1 zeigt die GUI die wir gemacht haben. Hier werden wir ein paar Hinweise geben, wie man unsere GUI benutzen kann:

- **Step 1.** Klicken auf den "Import Path" Knopf. Dieser Knopf wird den Pfad nach Benutzer's Wunsch führen.

- **Step 2.** Zum Erst muss man wählen, welches Bild plotten möchte. Hier gibt es zwei Optionen: links und rechts. Nach dem Wählen muss der Knopf "plot original image" geklickt werden.

- **Step 3.** Hier muss man einen Algorithmus auswählen. Wir haben drei Optionen: BM+SAD, BM+Census und GM.
- **Step 4.** Klicken auf den Knopf "Start Mapping".
- **Step 5.** Klicken auf den Knopf "3D Reconstruction".

Die Disparität wird nach dem Klicken auf "Start Mapping" geplottet bei der rechten Seite. Und die entsprechenden Prozesszeit, PSNR, R, T und p werden auch dargestellt. Nach dem Klicken auf "3D Reconstruction" kann man ein 3D Rekonstruktion Bild bekommen.

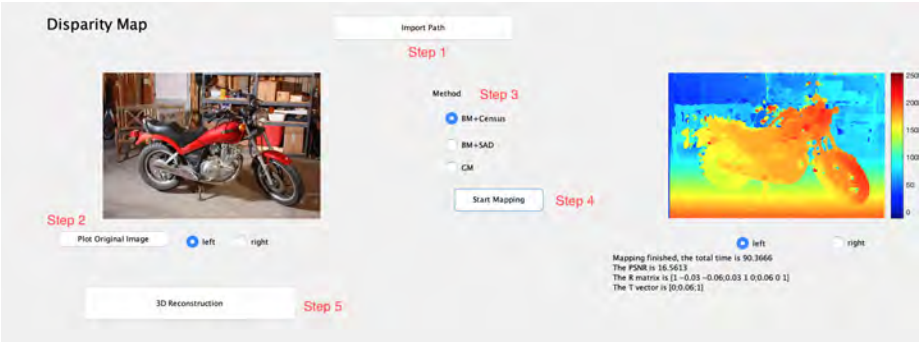


Figure 1: GUI Diagramm

3.2 Unittest

Diese Block Function funktioniert als drei Funktionen, um die Wert von R,T,D und toolbox bzw. tolerenz zu testen. Durch Classify_ Dokument werden die Funktionen check_toolboxes, check_variables und check_psnr definiert. Alle Variables werden in dem Diskurs des Attributs zugegeben. Drei Funktionen werden in der methodes Diskurs zugesetzen. Das Ergebnisprognose (Beurteilung) wird in Command_line als drei Diskus erscheinen.

3.3 Verification

Disese komponent ist für PSNR zu rechnen. Vor der Rechnung wird Graustufen-Bildnormalisierung der beiden Matrizen auf das Intervall [0; 255] nach die folgende Formel durchgeführt. D ist der Matrizen der Disparität und G vertret sich die Matrizen des Groundtruth.

$$D = (D - min(D(x:y)).*255./[max(D(x:y)) - min(D(x:y))] \quad (10)$$

$$G = (G - min(G(x:y)).*255./[max(G(x:y)) - min(G(x:y))] \quad (11)$$

Sehr häufig wird zur Bewertung der Bildqualität der "mittlere quadratische Fehler" (MSE). Man berechnet die Differenz jedes einzelnen Bildpunktes des

Originals zum gestörten Bild. MSE werden alle summiert und anschließend mit der Anzahl aller Bildpunkte gewichtet, indem durch diese geteilt wird. D(x,y) bezeichnet hier die Pixel des ursprunglichen Bildes, G(x,y) die des fehlerbehafteten Signals.

$$MSE(D,G) = \sum_{x=0}^{n1-1} \sum_{y=0}^{n2-1} [D(x,y) - G(x,y)]^2 / (n1 * n2 * dim) \tag{12}$$

Möchte man die maximal mögliche Leistung des Bildsignals im Verhältnis zur Störleistung der auftretenden Fehle betrachten, so erhält man den PSNR. Dieser wird üblicherweise in Dezibel (dB) angegeben und ist wie folgt definiert:

$$PSNR = 10 * log10(255^2 / MSE(D,G)) \tag{13}$$

4 Ergebnisse

Die Ergebnisse von alle Algorithmus werden in diesem Abschnitt dargestellt.

4.1 BM

Für dem BM haben wir ein paar Bilder als Test Sätze verwendet. Zum Erst werden die Disparitäten von linkem Bild und rechtem Bild mit der Hilfe von SAD Algorithmus mit- und ohne dem Mean Filtering Verfeinerungsprozess verglichen. Die Ergebnisse liegen in Figure 2.

Als Figure 2 dargestellt, ist es auffällig, dass die Ergebnisse mit der Mean Filtering Verfinerung glatter aussehen. Danach ist auch notwendig, die BM Algorithmus mit der Census Transform darzustellen und zu vergleichen. Figure 3 zeigt die Ergebnisse davon.

Es fällt uns auf dass die Ergebnisse mit der Mean Filtering Verfeinerungsmethode besser als die ohne Verfeinerung. Darüber hinaus sind die Ergebnisse mit Census Transform und Mean Filtering Verfeinerung besser als die mit SAD Algorithmus und Mean Filterung Verfeinerung. Wir haben auch beobachtet, dass die Salt-and-pepper Noises in den Ergebnisse ohne die Verfeinerung entstehen. Für diese Noises muss normalerweise eine Median Filteringsverfeinerung durchgeführt werden. Figure 4 zeigt die Ergebnisse davon.

Allerdings sieht man von Figure 3 und 4 gar keinen Unterschied von der Verwendung der Mean Filterung und Median Filtering. Deswegen müssen wir in den genauen Details, das Peak Signal-To-Noise Ratio, einmal eingucken und vergleichen. Figure 5 zeigt die PSNR nach der Verwendung von der Mean Filterung und Median Filtering.

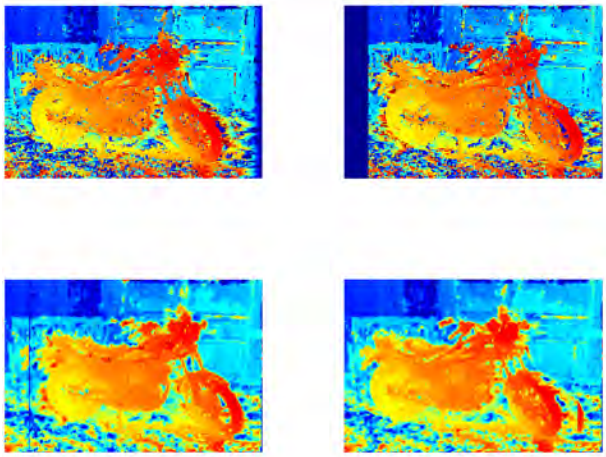


Figure 2: Ergebnises der Verwendung von BM + SAD. Die Bilder in der ersten Zeile sind die Ergebnisse ohne die Mean Filtering Verfeinerung. Die Bilder in der zweiten Zeile sind die Ergebnisse mit der Mean Filtering Verfeinerung. Das linke Bild in jeder Zeile ist die Disparität welche auf das Bild von rechter Kamera basiert (d.h. während der SAD oder Census Prozess ist die Schiebefenster auf dem rechten Bild fixiert und ist die Schiebefenster auf dem linken Bild verschoben). Und das rechte Bild in jeder Zeile ist die Disparität welche auf das Bild von linker Kamera basiert.

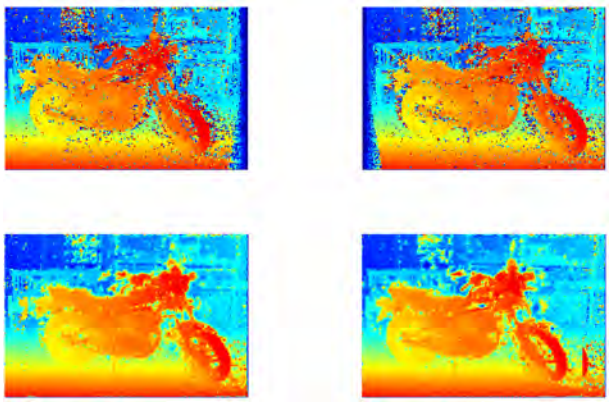


Figure 3: Ergebnises der Verwendung von BM + Census Transform + Mean Filtering. Das linke Bild ist die Disparität welche auf das Bild von rechter Kamera basiert. Und das rechte Bild ist die Disparität welche auf das Bild von linker Kamera basiert.

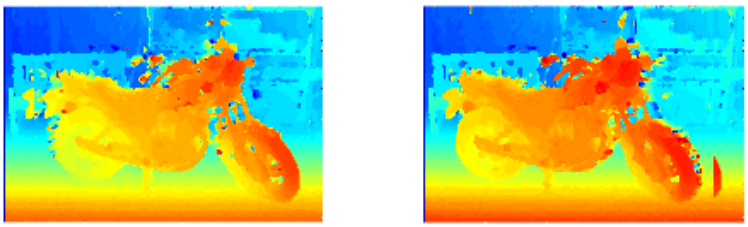


Figure 4: Ergebnises der Verwendung von BM + Census Transform + Median Filtering. Das linke Bild ist die Disparität welche auf das Bild von rechter Kamera basiert. Und das rechte Bild ist die Disparität welche auf das Bild von linker Kamera basiert.

BM_PSNR_meanfilter	16.3157
BM_PSNR_medianfilter	16.5613

Figure 5: PSNRs von der Vervendung der Mean Filtering und Median Filtering

4.2 GM

Für dem GM haben wir zum Erst ohne Verfeinerungsmethode benutzt. Figure 6 zeigt das Ergebnis davon. Es ist auffällig dass viele Pixel Geräusche auf dem Motorrad stehen. Desto haben wir entschieden, die Verfeinerungsprozesse durchzuführen.

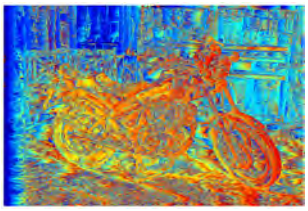
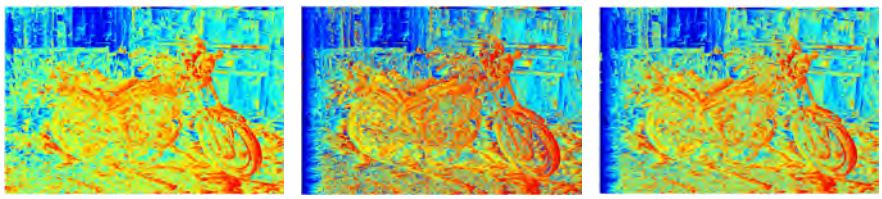


Figure 6: Ergebnises der Verwendung von GM ohne Verfeinerung

Für die Verfeinerungen haben wir mit der Mean Filtering, Median Filtering und Gaussian Filtering verwendet. Figure 7 zeigt die Ergebnisse von allen Verwendung der Filters. Es fällt uns auf, dass die Verfeinerung mit der Mean Filtering von Figure 7(a) nicht so gut vergleicht mit dem BM, sogar einbisschen schlecht geworden. Außerdem kann man die auffällige Verbesserung der Verwendung mit Median Filtering auch nicht sehen, wenn wir die Figure 7(b) mit Figure 6 vergleichen. Das heißt, dass die Geräusche von Figure 6 sehr wahrscheinlich nicht Salt-and-pepper Noises, sondern Gaussian Noises sind. Deswegen haben wir weiter mit Gaussian Filter verwendet und das Ergebnis können gesehen werden in Figure 7(c).



(a) Mit Mean Filtering Verfeinerung (b) Mit Median Filtering Verfeinerung (c) Mit Gaussian Filtering Verfeinerung

Figure 7: Ergebnises der Verwendung von GM mit unterschiedlichen Verfeinerungsmethode

Allerdings ist es schon sehr schwer für männchlichen Augen die Ergebnisse von Median Filtering und Gaussian Filtering in Figure 7 zu unterscheiden. Deshalb haben wir weiter die PSNR Werte von den Figures 6, 7(b) und 7(c) vergliechen. Figure 8 zeigt die PSNR für diese Verfeinerungsmethde. Wir beobachten, dass die Verfeinerung mit der Gaussian Filtering den besten PSNR Wert beherrscht. Das heißt, in dem Fall von GM, das beste Ergebnis ist mit der Gaussian Filtering.

GM_PSNR_gaussian	12.1017
GM_PSNR_medianfilter	11.6577
GM_PSNR_original	10.2809

Figure 8: PSNRs der verschiedenen Verfeinerungsmethode

4.3 3D-Rekonstruktion

Von allen Disparitäten Ergebnisse können wir Schlussfolgerung ziehen, dass die BM Algorithmus mit Census Transform und Median Filtering die besten Disparitäten haben. Und deswegen in diesem Abschnitt möchte wir das Ergebnis von diesem Algorithmus einmal wiederaufbauen, nämlich die sogenannte 3D-Rekonstruktion. In Figure 9 befindet sich die 3D-Rekonstruktion von "motorcycle", und für die zusätzlichen Ergebnisse von den anderen Bilder kann man in Section A.5 finden.

4.4 Vergleichung und Diskussion

Wir vergleichen alle Ergebnisse zusammen. Es ist auffällig von Figure 2, 4 und 7(c), dass die visuelle Leistungen von unterschiedlichen Algorithmus folgende

Figure 9: 3D-Rekonstruktion von “motorcycle”

Beziehung haben:

$$\begin{aligned}&\textbf{Performance}_{BM+Census\,Transform+Median\,Filtering} > \\ &\textbf{Performance}_{BM+SAD+Mean\,Filtering} > \\ &\textbf{Performance}_{GM+Gaussian\,Filtering}\end{aligned} \tag{14}$$

Wenn wir die Prozesszeit erwägen, dann sind die Leitungen anders und wie folgende dargestellt:

$$\begin{aligned}&\textbf{Speed}_{BM+SAD+Mean\,Filtering} > \\ &\textbf{Speed}_{GM+Gaussian\,Filtering} > \\ &\textbf{Speed}_{BM+Census\,Transform+Median\,Filtering}\end{aligned} \tag{15}$$

Es ergebt sich darauf, dass die BM Algorithmus besser als die GM sind, insbesondere die BM mit Census Transform und Median Filtering. Aber die Prozess Geschwindigkeit von BM mit Census Transform und Median Filtering am langsamsten ist. Im Vergleich dazu ist die BM mit SAD und Mean Filtering den zweiten Platz von visueller Performance und den ersten Platz von Prozess Geschwindigkeit gewonnen. Das heißt, wenn man die Disparität schnell bekommen möchte, die eine genuge Performance hat, empfehlen wir dann die BM mit SAD und Mean Filtering. Aber diese Methode passt nicht an alle Fällen. Manchmal wegen den Beleuchtungsprobleme wird die linke Kamera und rechte Kamera zwei Bilder mit unterschiedlichen Beleuchtung bekommen. Außerdem existiert manchmal eine große homogene Fläche in dem Bild. In diesen Fälle haben wir gefunden, dass die BM mit Census Transform und Median Filtering die Probleme von Beleuchtung einfach auflösen kann. Deswegen, die Auswahl der unterschiedlichen Algorithmus kommt auf die aktuelle Situation an.

5 Conclusion

In unserer Arbeit ist das Erhalten der Disparitäten spielt eine wichtige Rolle. Für das Erhalten der Disparitäten haben wir zwei unterschiedlichen Methoden verwendet, die sogenannte BM und GM. BM ist eine lokale Block-Matching Algorithmus, und GM ist eine pixelweise globale Optimierungsalgorithmus. Bei BM haben wir SAD und Census Transform als Verarbeitungsmethoden und Mean Filtering und Median Filtering als Verfeinerungsmethoden verwendet, während bei GM die Verarbeitung nicht mehr notwendig ist und die Verfeinerungen eine wichtige Rolle spielen. Wir beobachten, dass die BM mit Census Transform und Median Filtering die besten Ergebnisse beherrscht. Aber die Prozesszeit ist am langsamsten. Figure 9 und 21(a) zeigen die 3D-Rekonstruktionen von dieser Methode. Im Vergleich dazu liegt die BM mit SAD und Mean Filtering in dem zweiten Platz und hat die schnellste Prozesszeit. Allerdings verhalten die BM mit SAD und Mean Filtering und GM sehr schlecht bei den Fälle, dass die Beleuchtung von beide Kamera unterschiedlich ist und es eine große homogene Fläche in dem Bild gibt. Bei diesen Fälle kann die BM mit Census Transform und Median Filtering gut verhalten.

A Zusätzliche Ergebnisse von mehreren Bilder

A.1 BM + SAD

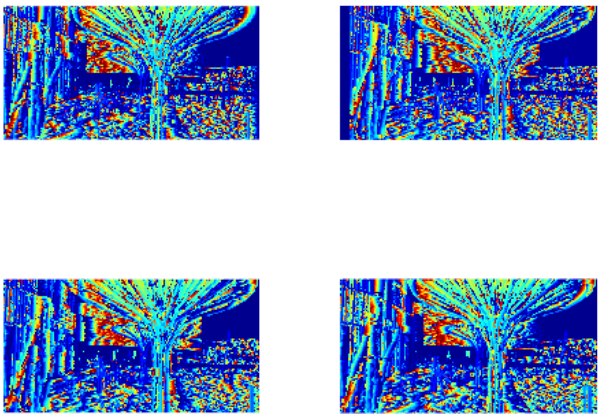


Figure 10: Ergebnises der Verwendung von BM + SAD für “playground”. Die Bilder in der ersten Zeile sind die Ergebnisse ohne die Mean Filtering Verfeinerung. Die Bilder in der zweiten Zeile sind die Ergebnisse mit der Mean Filtering Verfeinerung. Das linke Bild in jeder Zeile ist die Disparität welche auf das Bild von rechter Kamera basiert. Und das rechte Bild in jeder Zeile ist die Disparität welche auf das Bild von linker Kamera basiert.

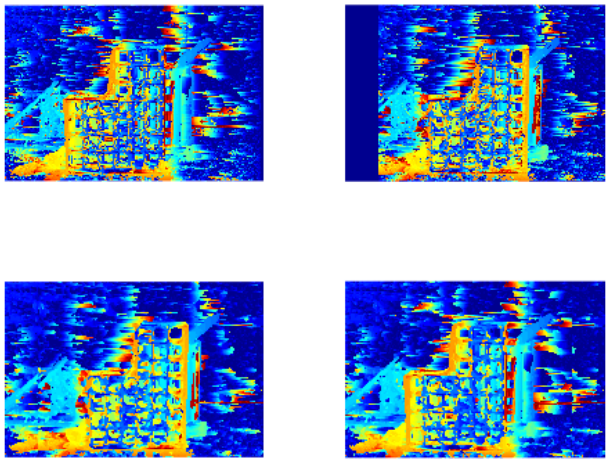


Figure 11: Ergebnises der Verwendung von BM + SAD für “sword”. Die Bilder in der ersten Zeile sind die Ergebnisse ohne die Mean Filtering Verfeinerung. Die Bilder in der zweiten Zeile sind die Ergebnisse mit der Mean Filtering Verfeinerung. Das linke Bild in jeder Zeile ist die Disparität welche auf das Bild von rechter Kamera basiert. Und das rechte Bild in jeder Zeile ist die Disparität welche auf das Bild von linker Kamera basiert.

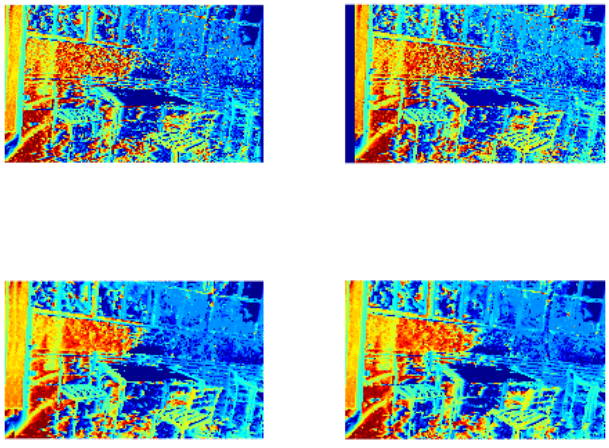


Figure 12: Ergebnises der Verwendung von BM + SAD für “terrace”. Die Bilder in der ersten Zeile sind die Ergebnisse ohne die Mean Filtering Verfeinerung. Die Bilder in der zweiten Zeile sind die Ergebnisse mit der Mean Filtering Verfeinerung. Das linke Bild in jeder Zeile ist die Disparität welche auf das Bild von rechter Kamera basiert. Und das rechte Bild in jeder Zeile ist die Disparität welche auf das Bild von linker Kamera basiert.

A.2 BM + Census Transform + Mean Filtering

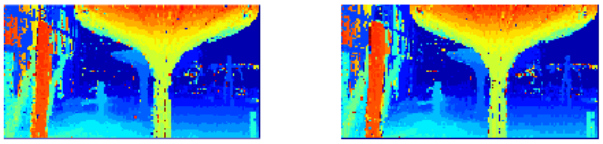


Figure 13: Ergebnises der Verwendung von BM + Census Transform + Mean Filtering für “playground”. Das linke Bild ist die Disparität welche auf das Bild von rechter Kamera basiert. Und das rechte Bild ist die Disparität welche auf das Bild von linker Kamera basiert.

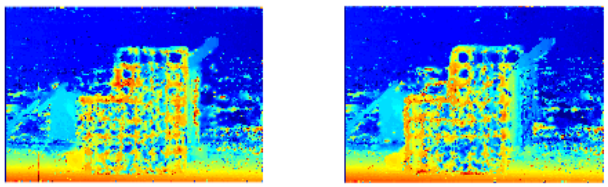


Figure 14: Ergebnises der Verwendung von BM + Census Transform + Mean Filtering für “sword”. Das linke Bild ist die Disparität welche auf das Bild von rechter Kamera basiert. Und das rechte Bild ist die Disparität welche auf das Bild von linker Kamera basiert.

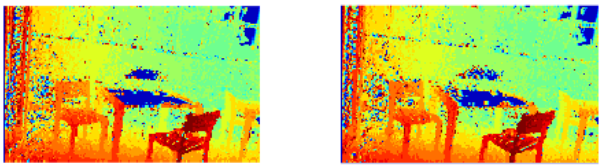


Figure 15: Ergebnises der Verwendung von BM + Census Transform + Mean Filtering für “terrace”. Das linke Bild ist die Disparität welche auf das Bild von rechter Kamera basiert. Und das rechte Bild ist die Disparität welche auf das Bild von linker Kamera basiert.

A.3 BM + Census Transform + Median Filtering

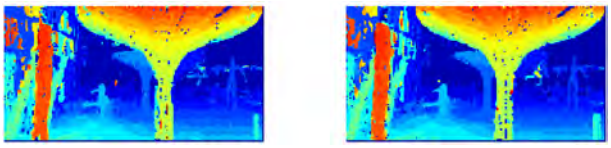


Figure 16: Ergebnises der Verwendung von BM + Census Transform + Median Filtering für “playground”. Das linke Bild ist die Disparität welche auf das Bild von rechter Kamera basiert. Und das rechte Bild ist die Disparität welche auf das Bild von linker Kamera basiert.

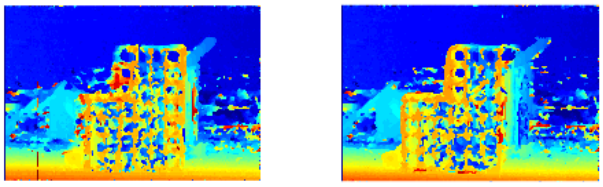


Figure 17: Ergebnises der Verwendung von BM + Census Transform + Median Filtering für “sword”. Das linke Bild ist die Disparität welche auf das Bild von rechter Kamera basiert. Und das rechte Bild ist die Disparität welche auf das Bild von linker Kamera basiert.

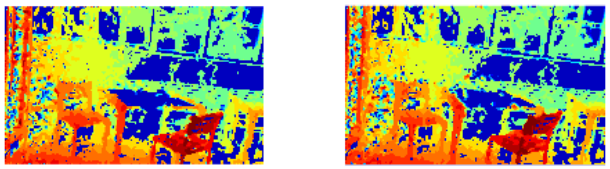


Figure 18: Ergebnises der Verwendung von BM + Census Transform + Median Filtering für “terrace”. Das linke Bild ist die Disparität welche auf das Bild von rechter Kamera basiert. Und das rechte Bild ist die Disparität welche auf das Bild von linker Kamera basiert.

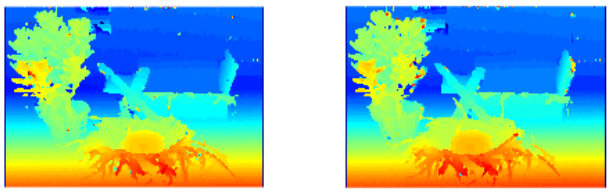
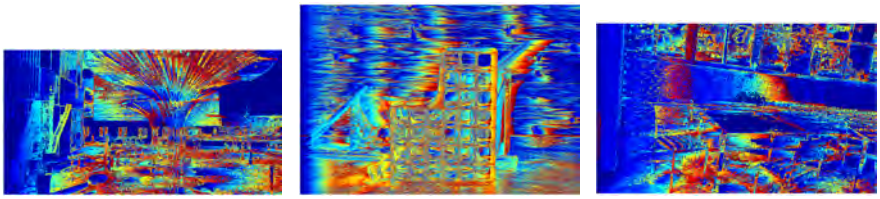


Figure 19: Ergebnises der Verwendung von BM + Census Transform + Median Filtering für "chair". Das linke Bild ist die Disparität welche auf das Bild von rechter Kamera basiert. Und das rechte Bild ist die Disparität welche auf das Bild von linker Kamera basiert.

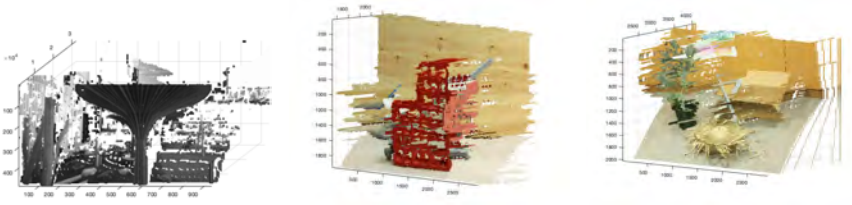
A.4 GM ohne Nachverarbeitung



(a) Playground (b) Sword (c) Terrace

Figure 20: Ergebnises der Verwendung von GM ohne Verfeinerung für unterschiedlichen Bilder

A.5 3D-Rekonstruktion



(a) 3D-Rekonstruktion von "playground" (b) 3D-Rekonstruktion von "sword" (c) 3D-Rekonstruktion von "chair"

Figure 21: 3D-Rekonstruktionen für die anderen Bilder

References

[1] Xing Mei, Xun Sun, Mingcai Zhou, Shaohui Jiao, Haitao Wang, and Xiaopeng Zhang. On building an accurate stereo matching system on graphics

hardware. In *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pages 467–474. IEEE, 2011.

[2] Jianbo Jiao, Ronggang Wang, Wenmin Wang, Shengfu Dong, Zhenyu Wang, and Wen Gao. Local stereo matching with improved matching cost and disparity refinement. *IEEE MultiMedia*, 21(4):16–27, 2014.

[3] Jaeryun Ko and Yo-Sung Ho. Stereo matching using census transform of adaptive window sizes with gradient images. In *2016 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA)*, pages 1–4. IEEE, 2016.

[4] Daniel Scharstein and Richard Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International journal of computer vision*, 47(1-3):7–42, 2002.