Institut für Informationsverarbeitung (TNT)
Appelstraße 9a, 30167 Hannover
www.tnt.uni-hannover.de

Prof. Dr.-Ing. Bodo Rosenhahn
Felix Winkler, M.Sc.

WiSe 2022/23 **14.12.2022**

Praktische Übung: Matlab für die medizinische und industrielle Bildinterpretation

Versuch 8: Disparitätsschätzung

Aufgabe 1 - Disparität und Distanzmaße für Bilder

Bei Kameras kann die Tiefe (Disparität) relativ einfach in einem Stereobild geschätzt werden, wenn diese rektifiziert sind, d.h. die Kameras liegen auf einer Basislinie und sind parallel ausgerichtet. Die geschätzte Tiefe ist nur relativ, es kann also unterschieden werden, was näher oder weiter entfernt ist, es kann aber keine exakte Entfernung bestimmt werden. Zur Schätzung wird zu jedem Bildpunkt in einem Bild der korrespondierende Punkt im zweiten Bild gesucht. Dieser liegt auf der Epipolarlinie. In rektifizierten Bildern entsprechen die Epipolarlinien den Bildzeilen. Die Tiefeninformation ergibt sich aus dem Abstand der korrespondierenden Punkte im Bild. Je größer der Abstand, desto näher ist das entsprechende Objekt.
Im Folgenden soll die Tiefe in rektifizierten Kameras geschätzt werden.

a) Implementieren Sie die Funktion `myDisparity.m`, in der eine einfache Disparitätsschätzung vorgenommen werden soll, indem zu jedem Punkt im linken Bild der entsprechende Punkt im rechten Bild gesucht wird.

   Die Bilder sind rektifiziert, es genügt also entlang der entsprechenden Bildzeile zu suchen. Zudem ist die Anordnung der Kameras bekannt: Die Punkte des linken Bildes können sich im rechten Bild nur weiter links befinden.

   Zur weiteren Reduktion des Suchraums dient der Parameter `maxDisparity`. Dieser schränkt den Suchraum ein und legt somit den kleinstmöglichen, detektierbaren Abstand zur Kamera fest.

Tipps zur Implementierung:

- Das Problem lässt sich elegant lösen, indem das rechte Bild schrittweise nach links "geschoben" wird und der Abstand zwischen allen Punkten in den Bildern bestimmt wird. Die Disparität an der Stelle (i, j) ist die Verschiebung k, die den kleinsten Abstand generiert.
- Eine Verschiebung nach links lässt sich einfach über eine Faltung mit dem Filterkern $f = [0, 0, 1]$ umsetzen.

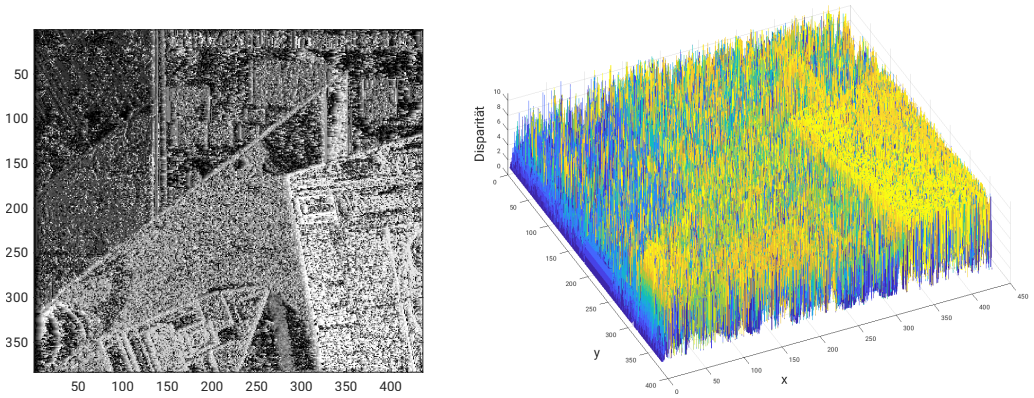

Abbildung 1: Beispielhafte Ergebnisse der einfachen Disparitätschätzung aus a)

b) Durch Bildrauschen werden korrespondierende Punkte im Bild nicht exakt gleich aussehen. Außerdem können mehrere ähnliche Punkte das Finden der richtigen Korrespondenzen erschweren.

Erweitern Sie Ihre Funktion, sodass nicht nur einzelne Bildpunkte vergleichen werden, sondern größere Bildausschnitte (eine Art Sliding Window). Fügen Sie einen weiteren Parameter `windowSize` hinzu, der die Größe der Bildausschnitte beschreibt. Verschiedene Distanzmaße sind möglich, um die Ähnlichkeit der Ausschnitte I_1 und I_2 zu vergleichen. Implementieren Sie

- Sum of Absolute Differences:

$$SAD(I_1, I_2) = \sum_{i,j} |I_1(i, j) - I_2(i, j)|$$

- Sum of Squared Differences:

$$SSD(I_1, I_2) = \sum_{i,j} (I_1(i, j) - I_2(i, j))^2$$

- Normalized Cross Correlation:

$$NCC(I_1, I_2) = 1 - \frac{cov^2(I_1, I_2)}{var(I_1)var(I_2)} = 1 - \frac{\left(\sum_{i,j} \left(I_1(i, j) - \bar{I}_1\right) \left(I_2(i, j) - \bar{I}_2\right)\right)^2}{\sum_{i,j} \left(I_1(i, j) - \bar{I}_1\right)^2 \sum_{i,j} \left(I_2(i, j) - \bar{I}_2\right)^2},$$

wobei $\bar{I} = \frac{1}{w^2} \sum_{i,j} I(i, j)$ den Mittelwert des Sliding Windows bezeichnet.

Hinweis: Die Abstandsmaße lassen sich ebenfalls über Faltungen darstellen, z.B.:

$$SAD(I_1, I_2) = |I_1 - I_2| * 1_{w \times w},$$

wobei w der Fenstergröße entspricht.

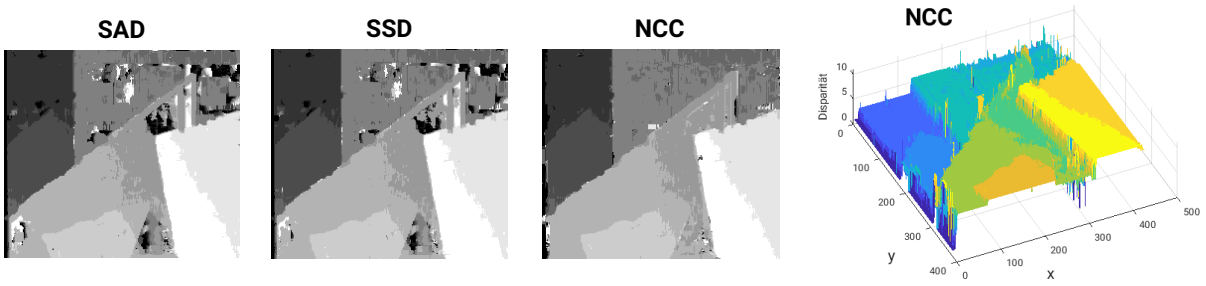

Abbildung 2: Beispielhafte Schätzungen mit den verschiedenen Distanzmaßen aus b)

Aufgabe 2 - Dynamic Time Warping

Die Tiefenschätzung aus Aufgabe 1 ist recht verrauscht. Dynamic Time Warping kann das Problem lösen, indem die Korrespondenzen global für jede Bildzeile bestimmt werden.

a) Vervollständigen Sie die Funktion `DTW.m`, in der Sie eine Variante des Dynamic Time Warping umsetzen, die sich an dem Paper "A Maximum Likelihood Stereo Algorithm" orientiert. Für den Occlusion-Strafterm kann eine einfachere Variante gewählt werden, bei der die aktuellen Kosten lediglich gewichtet werden, d.h. `Occlusion = weight * C(i, j)`.

   Testen Sie Ihre Funktion zunächst an einfachen Vektoren (`DTW_test.m`).

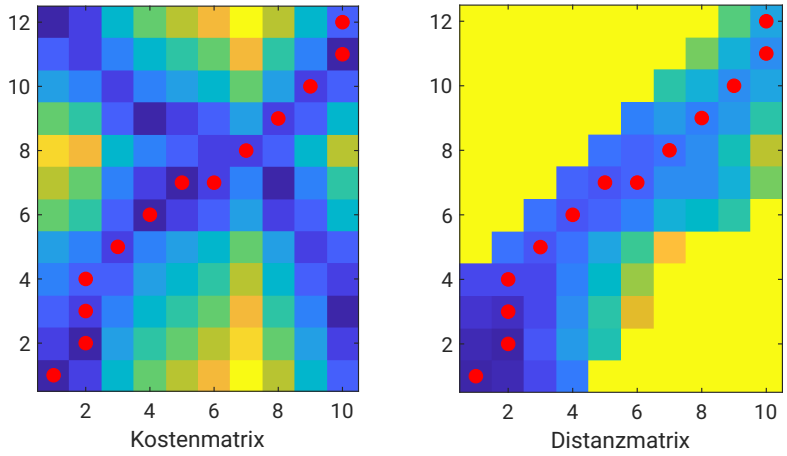

Abbildung 3: Ausgabe von `DTW_test.m`

b) Nutzen Sie ihre DTW-Funktion anschließend zur Disparitätschätzung (`DTW_disparity.m`), indem die Bildzeilen einzeln gewarped werden.

Tipp: Zur Beschleunigung sollte wiederum der Parameter `maxDisparity` genutzt werden, sodass die innere `for`-Schleife nicht über das komplette Bild läuft. Nicht berechnete Bereiche verbleiben dann bei unendlichen Kosten.

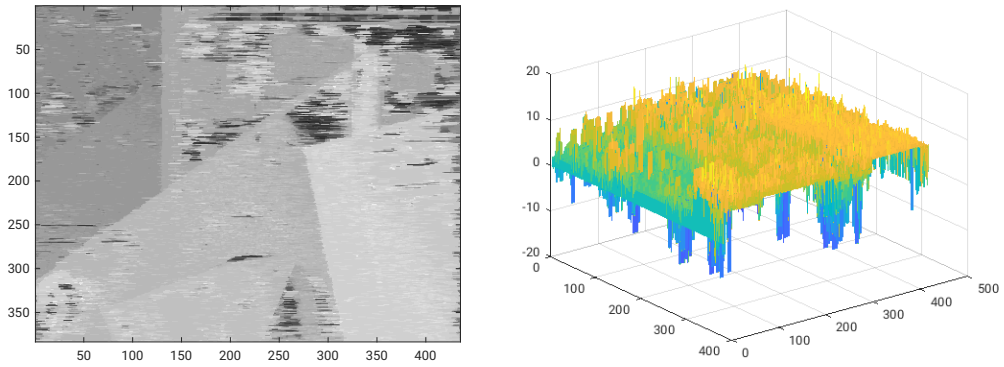

Abbildung 4: Ausgabe von `DTW_disparity.m` (zeilenweiser Vergleich)

c) Falls nicht bereits geschehen, erweitern Sie Ihre Funktion derart, dass wie in Aufgabe 1 nicht nur einzelne Bildpunkte, sondern auch größere Bereiche verglichen werden können. Hierzu muss lediglich die Berechnung der Kostenmatrix angepasst werden. Die Bildausschnitte werden als eine Menge von Spaltenvektoren übergeben. Passen Sie dazu den Parameter `windowSize` in `DTW_disparity.m` an.

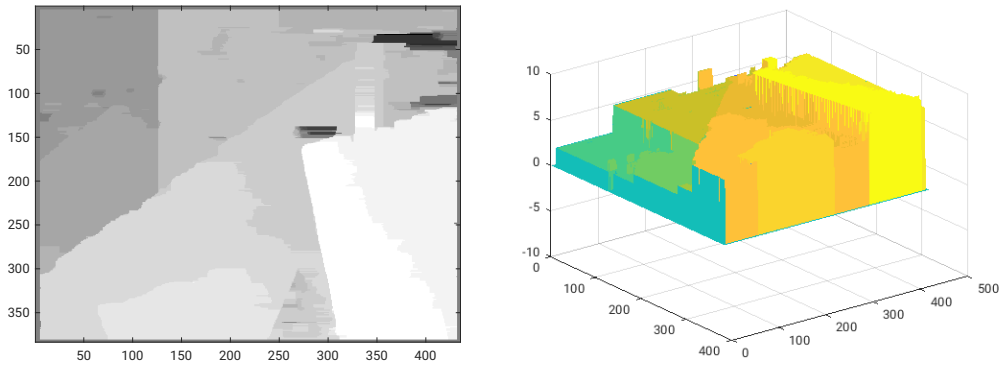

Abbildung 5: Ausgabe von `DTW_disparity.m` (Vergleich mehrerer Zeilen)

Kontrollfragen

a) Nennen Sie Unterschiede zwischen den Distanzmaßen aus Aufgabe 1 b).

b) Warum ist das Ergebnis der Disparitätsschätzung aus Aufgabe 1 teilweise stark verrauscht?

c) Welche Disparität erhalten verdeckte Bildpunkte in Aufgabe 1 und Aufgabe 2?