

Aufgabenblatt 4

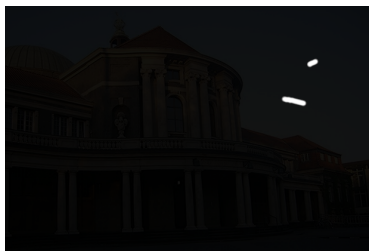
Übung zu Einführung in die Bildverarbeitung

Laszlo Korte, Thang Le, Juri Schalück, Maris Hillemann, Tim Rolff, Christian Wilms
SoSe 2023

Ausgabe: 12. Mai 2023 - **Abgabe bis: 23. Mai 2023, 10:00**

Abgabe per Moodle

Juri Schalück (Gruppe 2)	0schalue@informatik.uni-hamburg.de
Thang Le (Gruppe 3)	phuoc.thang.le@uni-hamburg.de
Laszlo Korte (Gruppe 4)	9korte@informatik.uni-hamburg.de
Maris Hillemann (Gruppe 5, 6)	maris.nathanael.hillemann@studium.uni-hamburg.de
Tim Rolff (Gruppe 1)	tim.rolff@uni-hamburg.de
Christian Wilms (Gruppe 1)	christian.wilms@uni-hamburg.de



(a)



(b)



(c)

Abbildung 1: Bilder für die Bildverbesserung in Aufgabe 1.

Aufgabe 1 — Bildverbesserung - 5+5+15=25 Punkte - Programmieraufgabe

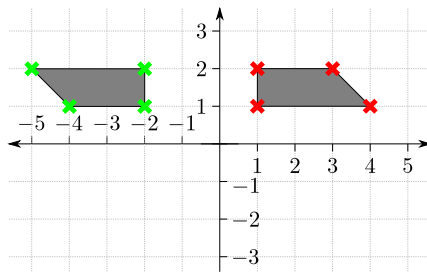
Erlaubte (Sub-)Pakete: `numpy`, `skimage.io`, `matplotlib`, `math`

Abbildung 1 zeigt drei Bilder, die im Rahmen dieser Aufgabe mit Hilfe von jeweils einer Intensitätstransformation so verbessert werden sollen, dass eine vorgegebene Information einfacher aus den Bildern ausgelesen werden kann. Überlegt euch dazu zunächst welche Intensitätstransformation passend erscheint (mit kurzer Begründung) und implementiert diese jeweils in Python. Die in Abbildung 1 dargestellten Bilder findet ihr im Moodle als `bildverbesserung.zip`.

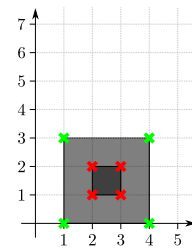
Hinweis 1: Für manche Bilder kann es sich anbieten, die Pixelwerte vom Wertebereich $0, \dots, 255$ auf den Bereich $0, \dots, 1$ zu ändern.

Hinweis 2: Die Ergebnisse werden nicht perfekt werden!

1. Verbessert den Kontrast im Bild und ermittelt, was in Abbildung 1a durch die zwei Bildfehler verdeckt wird.
2. Verändert das Bild in Abbildung 1b so, dass in etwa der Bereich des Helms und des Anzugs (grau Fläche in der Mitte und unten rechts) schwarz ist und der Rest des Bildes seine Graufärbung behält. Einzelne andere Pixel können ebenfalls verfärbt werden.
3. Invertiert das Bild und erhöht den Kontrast des Hubschraubers in Abbildung 1c in Bezug auf seine Umgebung (Himmel). Lasst dabei die Helligkeitswerte der Vegetation am unteren und rechten Bildrand (nahezu) unverändert.



(a)



(b)

Abbildung 2: Zwei Szenarien, in denen jeweils eine Transformationsmatrix auf die Eckpunkte eines Vierecks (rote Kreuze) angewendet wurde, um ein neues Viereck zu erzeugen (grüne Kreuze).

Aufgabe 2 — Geometrischen Transformationen - $5+5+7+7+8+8=40$ Punkte - Theorieaufgabe
Diese Aufgabe widmet sich geometrischen Transformationen.

1. Gebt zu den zwei Szenarien in Abbildung 2 die jeweilige Transformationsmatrix an. In den zwei Szenarien sollen die Transformationen jeweils auf die Eckpunkte der Vierecke mit den roten Kreuzen angewendet werden, sodass jeweils das veränderte Vierecke (grüne Kreuze) entsteht. Bei Sequenzen von Transformationen sollen die einzelnen Transformationsmatrizen und die Berechnung der finalen, kombinierten Transformationsmatrix gegeben werden.

Tip: Wendet einmal eure erzeugte Transformationsmatrizen auf die entsprechenden Punkte an und betrachtet das Ergebnis. Stimmt es?

2. Interpretiert die beiden folgenden Transformationsmatrizen und beschreibt kurz welche einzelnen Transformationen (inklusive Parameter wie Rotation um 180°) in welcher Reihenfolge hier beschrieben werden.

Tip: Es gibt tlw. mehrere Lösungen.

a)

$$\begin{bmatrix} 0 & -1 & -1 \\ 1 & 0 & 3 \\ 0 & 0 & 1 \end{bmatrix}$$

b)

$$\begin{bmatrix} 1 & 0 & 1 \\ 0 & 4 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

3. Beweist oder widerlegt mit einem Gegenbeispiel die Kommutativität folgender geometrischer Transformationen bei der Anwendung auf Punkte $\begin{bmatrix} x \\ y \end{bmatrix} \in \mathbb{R}^2$.

a) Scherung und Translation

b) Rotation und Rotation mit unterschiedlichen Faktoren α und β .

Hinweis: Eine Rotation um den Winkel α lässt sich wie folgt beschreiben:

$$\begin{bmatrix} \cos(\alpha) & -\sin(\alpha) & 0 \\ \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Aufgabe 3 — Bildskalierung und Interpolation - $25+5+5=35$ Punkte - Programmieraufgabe

Erlaubte (Sub-)Pakete: `numpy`, `skimage.io`, `matplotlib`, `itertools`

Durch die geometrische Transformation von Bildern, bspw. die uniforme Skalierung um einen Faktor s , werden Pixelwerte an Koordinaten des Ursprungsbildes benötigt, die zwischen den eigentlichen Koordinaten liegen. So kann es etwa vorkommen, dass ein Wert an der Koordinate $(0.2, 0.8)$ benötigt wird. Da es in Bildern aber nur ganzzahlige Koordinaten gibt, muss der Pixelwert an der Koordinate $(0.2, 0.8)$ interpoliert werden. Dies soll in dieser Aufgabe mit der nearest neighbor Interpolation im Rahmen der Bildskalierung umgesetzt werden. Bei der nearest neighbor Interpolation wird, wie in Abbildung 3 dargestellt, der Pixelwert der Koordinate genommen, die der benötigten Koordinate (bspw. $(0.2, 0.8)$) am nächsten liegt. In diesem Beispiel wäre dies die Koordinate $(0, 1)$.

1. Schreibt eine Python-Funktion, die ein Bild um einen gegebenen Faktor, der nicht ganzzahlig sein muss, uniform skaliert. Nutzt dazu die nearest neighbor Interpolation. Testet nun eure Skalierung am Bild `tv.png` aus dem Moodle.
Hinweis 1: Rechnet die Koordinaten des Zielbildes in die Koordinaten des Ausgangsbildes zurück und interpoliert dort.
Hinweis 2: Die Funktion `itertools.product()` kann nützlich sein, um aus den einzelnen x - bzw. y -Koordinaten alle Koordinaten des Bildes zu erzeugen.
2. Werden durch das Skalieren von Bildern mit einem Faktor $s > 1$ Informationen gewonnen?
3. Was passiert, wenn ihr das Bild zunächst um die Hälfte verkleinert ($s = 0.5$) und anschließend wieder auf die vorherige Größe vergrößert ($s = 2$)? Ist das Ergebnis identisch zum Ursprungsbild?

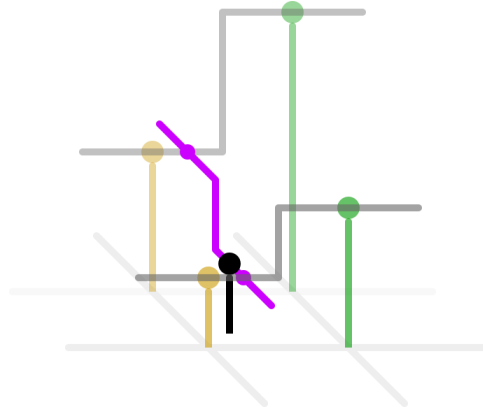


Abbildung 3: Visualisierung der nearest neighbor Interpolation. Der Grauwert an der schwarzen Koordinate, die nicht im Bild existiert, soll interpoliert werden. Die anderen vier Koordinaten existieren im Bild und sind die Basis für die Interpolation. Quelle: Wikimedia ¹

Zusatzaufgabe 4 — Rotation von Bildern - 30+5+5=40 Punkte - Programmieraufgabe

Erlaubte (Sub-)Pakete: `numpy`, `skimage.io`, `matplotlib`, `itertools`

Neben der Skalierung ist die Rotation um den Mittelpunkt des Bildes eine gebräuchliche geometrische Transformation auf Bildern.

1. Implementiert nun eine Funktion, welche die Bildrotation um einen beliebigen Winkel durchführt. Geht dabei davon aus, dass das Zielbild stets die Größe $\lfloor \sqrt{2}h \rfloor \times \lfloor \sqrt{2}w \rfloor$ hat für ein Originalbild der Größe $h \times w$. Nutzt zudem wieder die nearest neighbor Interpolation und setzt Pixel, deren Wert im Zielbild ihr nicht bestimmen könnt auf den nächsten vorhandenen Wert.
2. Wendet eure Funktion mit dem Winkel -45° und 90° auf das Bild `tv.png` an und vergleicht es mit dem Ergebnis der Funktion `skimage.transform.rotate()`. Achtet dabei darauf, dass ihr bei `skimage.transform.rotate()` die nearest neighbor Interpolation auswählt und das Zielbild ggf. größer sein darf als das Originalbild.
Hinweis: Euer Ergebnis wird etwas anders aussehen als das von `skimage.transform.rotate()`. Den schwarzen Rand im Ergebnis von `skimage.transform.rotate()` könnt ihr ignorieren.
3. Was passiert mit dem Bildinhalt, wenn ihr das Bild zunächst um $22,5^\circ$ und dann um $-22,5^\circ$ rotiert?

¹https://commons.wikimedia.org/wiki/File:Comparison_of_1D_and_2D_interpolation.svg ↗