

Aufgabenblatt 3

Übung zu Einführung in die Bildverarbeitung

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

Ausgabe: 5. Mai 2023 - **Abgabe bis: 15. Mai 2023, 10:00**

Abgabe per Moodle

Juri Schalück (Gruppe 2)	oschalue@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

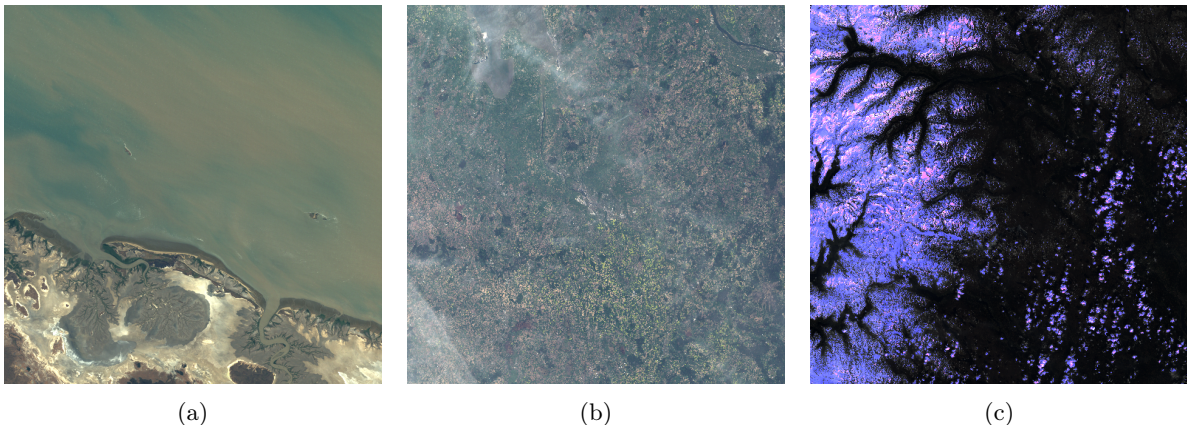


Abbildung 1: Echtfarben-Darstellungen von drei Landsat 5-Satellitenbildern.

Aufgabe 1 — Interpretation von Satellitenbildern - 20+3+3+4=30 Punkte - Programmieraufgabe

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

In der Fernerkundung werden verschiedene Indizes genutzt, um zu prüfen, welche Pixel eines Satellitenbildes etwa grüne Vegetation oder Wasser zeigen. Ein Beispiel ist der Vegetationsindex NDVI (normalized difference vegetation index), der für jedes Pixel eines Satellitenbildes berechnet wird. Der NDVI ist ein Indikator für lebende, grüne Vegetation. Für seine Berechnung werden die Intensitäten des roten Lichts (Red, Wellenlänge $0.63\mu\text{m} - 0.69\mu\text{m}$) und des nahen Infrarots (NIR, Wellenlänge $0.78\mu\text{m} - 0.90\mu\text{m}$) in einem Pixel benötigt. Mithilfe dieser Werte kann der NDVI für ein Pixel als

$$\text{NDVI} = \frac{(\text{NIR} - \text{Red})}{(\text{NIR} + \text{Red})}$$

berechnet werden. Ähnlich ist der Wasserindex NDWI (Normalized Difference Water Index) definiert. Dieser ist ein Indikator für Wasser und basiert auf den Intensitäten des nahen Infrarots und des kurzwelligen Infrarots (SWIR, Wellenlänge $2.09\mu\text{m} - 2.35\mu\text{m}$):

$$\text{NDWI} = \frac{(\text{NIR} - \text{SWIR})}{(\text{NIR} + \text{SWIR})}.$$

Beide Indizes produzieren Werte im Bereich von +1 bis -1. Höhere Werte deuten dabei auf grüne Vegetation bzw. Wasser hin, wohingegen mittlere und niedrige Werte gegen deren Vorliegen sprechen.

1. Implementiert die beiden Indizes und wendet sie auf die Satellitenbilder in Abbildung 1 an (s. `landsatBilder.zip` im Moodle). Für jedes der Bilder stehen euch dazu im `landsatBilder.zip` die 7 Bänder des jeweiligen Landsat 5-Satellitenbildes zur Verfügung. Die Bänder sind einzeln als Graustufenbilder gespeichert. Als Resultat sollt ihr die insgesamt sechs Ergebnisbilder mit `matplotlib` anzeigen.

Tipp: Um eine realistische Darstellung zu erhalten, passt bei `matplotlib.pyplot.imshow` die Parameter `vmin` und `vmax` auf den Wertebereich von NDVI bzw. NDWI an.

2. Analysiert das Satellitenbild in Abbildung 1a anhand der Indizes und des Echtfarbenbildes. Worum handelt es sich bei den dunklen Flächen unten im Bild und der hellen großen flachen Fläche? Gebt eine kurze Antwort mit Begründung (max. 3 Sätze).
3. Interpretiert nun das Satellitenbild in Abbildung 1b. Worum handelt es sich bei den größeren dunklen Bereichen? Zeigen diese Bereiche Seen, Gestein oder Kohleminen? Gebt eine kurze Antwort mit Begründung (max. 3 Sätze).
4. Zuletzt analysiert nun das Satellitenbild in Abbildung 1c. Worum handelt es sich bei der rosanen Fläche im Bild? Gebt auch eine Begründung an, warum dieses Objekt diese Farbe hat. Gebt eine kurze Antwort mit Begründung (max. 3 Sätze).

Aufgabe 2 — Bildinhalte kombinieren - $10+12+10+8(+20)=40(+20)$ Punkte - Programmieraufgabe

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

In der Vorlesung wurde das Mitteln über mehrere zeitlich versetzt aufgenommene Bilder einer identischen Szene als Möglichkeit zur Unterdrückung des Rauschens präsentiert. Diese Idee soll in dieser Aufgabe genutzt werden, um aus Einzelbildern, die verschiedene Zustände einer Szene zeigen, ein Gesamtbild zu erzeugen. Ein Beispiel dazu bietet das kombinierte Serienbild in Abbildung 2, das aus mehreren Einzelbildern besteht, die jeweils einen Bewegungszustand des Balls darstellen. Konkret geht es in dieser Aufgabe um die vier Einzelbilder des `kombibild.zip` aus dem Moodle, die zu einem Bild zusammengesetzt werden sollen.

1. Ladet die vier Einzelbilder des `kombibild.zip` aus dem Moodle in Python. Zunächst muss aus den vier Einzelbildern ein möglichst perfektes Hintergrundbild ohne Kronkorken erzeugt werden. Wenn man die Kronkorken als Rauschen annimmt, kann man nach dem Gesetz der großen Zahlen durch Mittelung über die Bilder das Rauschen, bzw. hier die Kronkorken, zumindest recht gut entfernen. Reichen vier Bilder dafür aus? Probiert es aus!
2. Überprüft das Ergebnis visuell und überlegt euch eine bessere aber sehr ähnliche Möglichkeit für das Verknüpfen der vier Pixel an jeweils einer Koordinate. Betrachtet dabei keine anderen Koordinaten wie etwa die der räumlichen Nachbarn! Das Ergebnis sollte nahezu perfekt aussehen.
3. Nutzt nun euer erzeugtes Hintergrundbild, um die veränderten Pixel in jedem der vier Einzelbilder zu ermitteln. Damit ihr wirkliche Veränderungen vom Grundrauschen unterscheiden könnt, empfiehlt sich die Nutzung eines Schwellenwerts für die Mindeststärke der Veränderung. Den Schwellenwert müsst ihr selbst ermitteln.
4. Ersetzt nacheinander für jedes der vier Bilder die veränderten Pixel im Hintergrundbild durch die entsprechenden Pixel des Einzelbildes. D.h. die Pixel, die sich zwischen dem Hintergrundbild und dem ersten Einzelbild verändert haben, werden im Hintergrundbild durch die entsprechenden Pixel des ersten Einzelbildes ersetzt usw. Speichert das Ergebnisbild ab. Welche Figur ergeben die zusammengesetzten Kronkorken nun?
5. **Zusatzaufgabe:** Nehmt selbst Einzelfotos oder ein Serienfoto auf und versucht euer Verfahren darauf anzuwenden. Achtet dabei auf eine möglichst ruhige Kamera und benutzt wenn möglich ein Stativ. Um ein Farbbild in ein Graustufenbild umzuwandeln, könnt ihr die Funktion `skimage.color.rgb2gray` nutzen und das Ergebnis mit 255 multiplizieren. Originelle Ergebnisse präsentieren wir gerne in der Übung!

Aufgabe 3 — Räumliche Operationen - $3+3+4+4+4+4+4+4=30$ Punkte - Theorieaufgabe

Schaut euch die acht folgenden Python-Funktionen an und beantwortet zu jeder Funktion, ob es sich dabei um eine Punktoperation, eine Nachbarschaftsoperation, eine geometrische Transformation oder eine globale Operation handelt. Gebt jeweils eine kurze Begründung. Nehmt an, dass das Bild eine

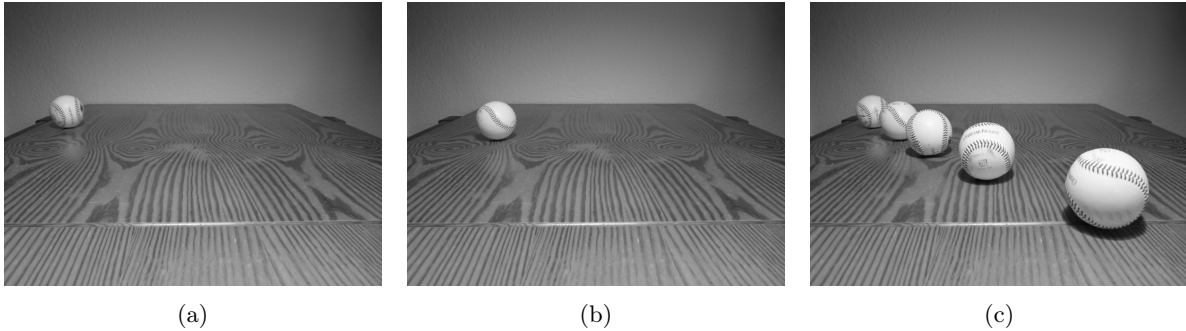


Abbildung 2: Die ersten beiden Einzelbilder (a und b) eines kombinierten Serienbildes (c), das aus der Kombination von fünf Einzelbildern besteht.

Auflösung von $n \times n$ hat.

Hinweis: Die Nachbarschaft eines Pixels muss nicht unbedingt alle bzw. nur die acht direkt benachbarten Pixel enthalten.

```
def a(img):
    return img - np.median(img)

def b(img):
    return (np.abs(img) - img) / 2

def c(img):
    return np.minimum(img, 255 - img)

def d(img):
    return img - img @ img

def e(img):
    return np.transpose(img)

def f(img):
    return img[1:-1,1:-1] + img[:-2,:-2] + img[2:,2:]

def g(img):
    result = np.zeros_like(img, dtype=np.int)
    for x in range(img.shape[0]):
        for y in range(img.shape[1]):
            result[x,y] = np.sum(img[:x,:y])
    result[:,:] = result[0,0]
    return result

def h(img):
    result = np.copy(img)
    for x in range(img.shape[0]-1):
        for y in range(img.shape[1]-1):
            a = img[x,y]
            b = img[x,y+1]+img[x+1,y+1]+img[x+1,y]
            if a>b and False:
                result[x,y]=a
            else:
                result[x,y]=b
    return result
```

Zusatzaufgabe 4 — Findet die Fehler - 5+5+20=30 Punkte - Programmieraufgabe

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

Das Finden von Fehlern in der manipulierten Variante eines Originalbilds ist ein beliebtes Rätsel. Im Rahmen dieser Aufgabe soll dieses Problem nun strukturiert angegangen und mithilfe eines Python-Skripts gelöst werden. Dazu findet ihr im Moodle zwei Varianten eines Bilds, die eine ohne Veränderungen, die andere mit Veränderungen.

1. Ermittelt zunächst ein Bild, das die Veränderungen zwischen den beiden Bildern zeigt. Verknüpft dazu die Bilder mit einem sinnvollen arithmetischen Operator und wandelt die Bilder vorher von RGB in ein Graustufenbild um.
2. Wandelt das Ergebnis anschließend in ein Binärbild um, das veränderte Pixel als Vordergrund und unveränderte Pixel als Hintergrund beinhaltet. Wenn ihr euch das Bild nun anzeigen lasst, könnt ihr zwar die Positionen der veränderten Bereiche selbst ermitteln und die Anzahl zählen, aber das soll nun ebenfalls der Computer machen.
3. Erweitert daher euer Skript, um die Funktionalität des Zählens der veränderten Bereiche. Dabei ist jeder zusammenhängende Bereich an Pixeln, der sich verändert hat, als ein solcher Bereich zu zählen.
 - a) Ermittelt dazu zunächst alle Koordinaten von Pixeln, die sich verändert haben.
 - b) Startet nun bei einem beliebigen Pixel und prüft ob dessen Nachbarn auch verändert wurden. So könnt ihr euch durch die Menge der Vordergrundkoordinaten hangeln und iterativ die zusammenhängenden, veränderten Bereiche vollständig aufspüren.
 - c) Welche Anzahl an veränderten Bereichen ermittelt euer Skript?

Hinweis 1: Nutzt nur die oben angegebenen Pakete und Subpakete, sonst gibt es keine Punkte!

Hinweis 2: Achtet beim Durchlaufen der Vordergrundkoordinaten auf sinnvolle Abbruchkriterien, um endlose Schleifen zu vermeiden.