

Übung 2: Farben

Übung von: Remo Schwarzenruber

Detektion mit Farben

In unserem SkinApp Projekt befassen wir uns mit der Erkennung von Hautkrankheiten, zum Beispiel auf Händen. Als erstes muss dabei die Handfläche vom Hintergrund getrennt werden. Damit dies einfacher geht, haben wir eine Photobox entwickelt, die die Hände vor grünem Hintergrund aufnimmt. Sie sollen nun untersuchen, ob es damit tatsächlich einfacher ist die Handflächen oder Handrückseiten zu finden.

```
In [3]: # OpenCV needs to be included first
import cv2
import numpy as np

# for displaying images in jupyter
import matplotlib as mpl
from matplotlib import pyplot as plt
%matplotlib inline
# resolution of images in figure
mpl.rcParams['figure.dpi']= 150
# figure size in inches
mpl.rcParams["figure.figsize"] = [10,5]
```

Aufgabe 1: Hand Segmentierung

Als erstes sollen sie versuchen die Handflächen mittels der Farbe der Haut zu detektieren. Dazu ist es möglicherweise hilfreich das Bild in einer anderen Farbraum zu transferieren. Für Hautsegmentierung könnte sich zum Beispiel der YCrCb oder der HSV Farbraum eignen. Stellen Sie das Bild in diesen Farträumen dar und suchen Sie dann entsprechende Schwellwerte um eine binäre Maske der Hand zu erstellen. Für Schwellwerte stehen in OpenCV eigentlich die Funktionen `cv.threshold()` und `cv2.inRange()` zur Verfügung, für einfache Fälle ist oft auch ein Vergleich direkt in numpy möglich.

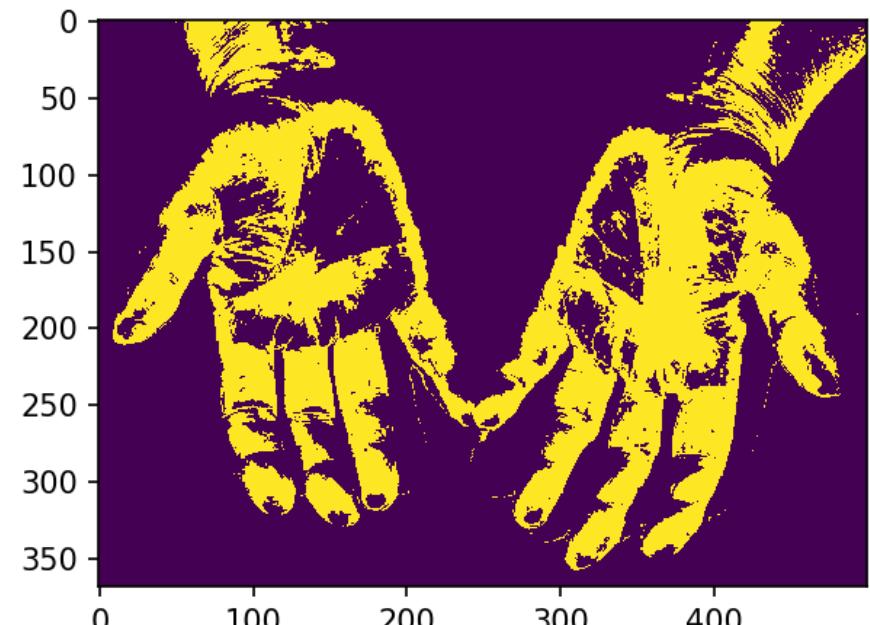
```
In [4]: image_grey_back = cv2.imread('../data/hand_grey_small.png')
image_grey_back = cv2.cvtColor(image_grey_back, cv2.COLOR_BGR2RGB)

# simple threshold in RGB
min = np.array([150, 0, 0])
max = np.array([255, 140, 150])

skin_image = cv2.inRange(image_grey_back, min, max)

plt.subplot(1, 2, 1)
plt.imshow(image_grey_back)
plt.subplot(1, 2, 2)
plt.imshow(skin_image)
```

Out[4]: <matplotlib.image.AxesImage at 0x121b37438>



Konvertieren sie nun das Bild in ein anderes Farbsystem und versuchen sie da ein besserer Schwellwert zu bekommen.

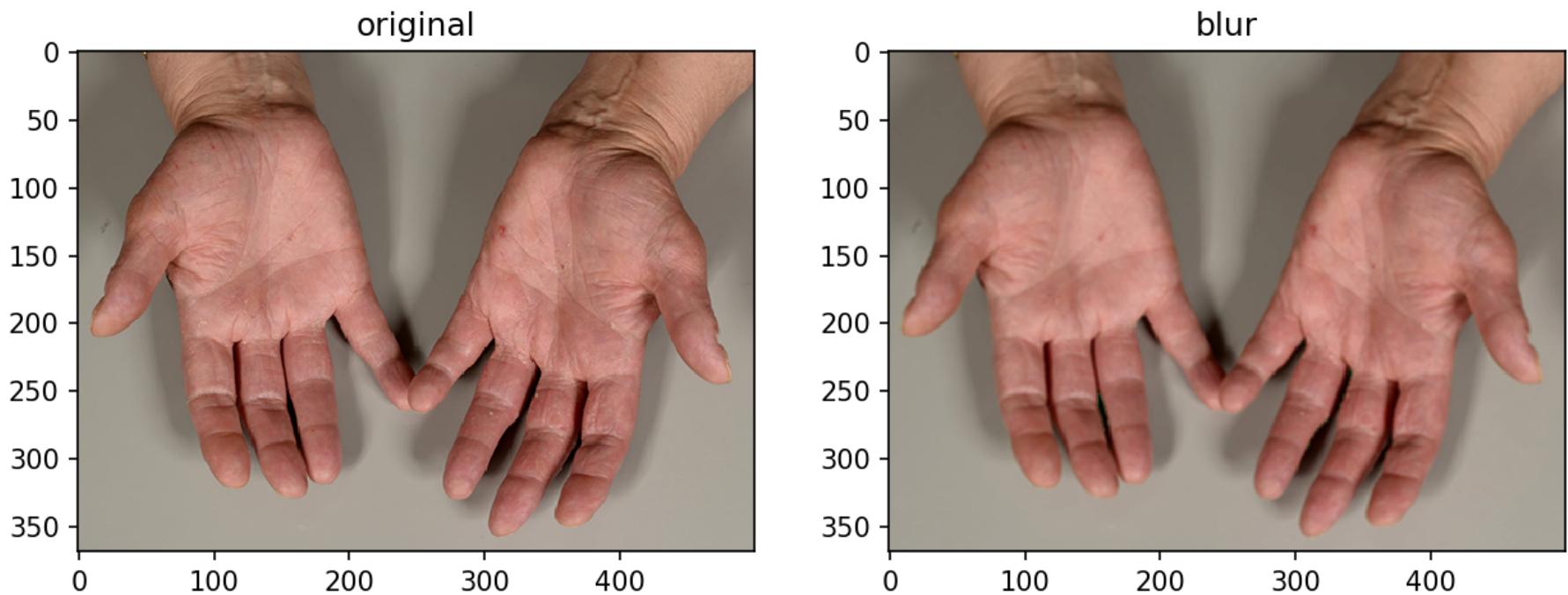
Aufgabe 1 mit HSV Farbraum

```
In [5]: image_grey_back = cv2.imread('../data/hand_grey_small.png')
image_grey_back = cv2.cvtColor(image_grey_back, cv2.COLOR_BGR2RGB)
image_grey_back_hsv = cv2.cvtColor(image_grey_back, cv2.COLOR_BGR2HSV)

# Adjust image a little bit (Anti-Aliasing)
image_grey_back_hsv_blur = cv2.blur(image_grey_back_hsv, (3,3))
plt.subplot(1, 2, 1)
plt.title("original")
plt.imshow(image_grey_back)

plt.subplot(1, 2, 2)
plt.title("blur")
plt.imshow(cv2.cvtColor(image_grey_back_hsv_blur, cv2.COLOR_HSV2BGR))
```

Out[5]: <matplotlib.image.AxesImage at 0x1221a22e8>



```
In [6]: # Detect color- and shadow-areas of the hand
lower = np.array([30, 20, 0], dtype=np.uint8)
upper = np.array([110, 70, 254], dtype=np.uint8)

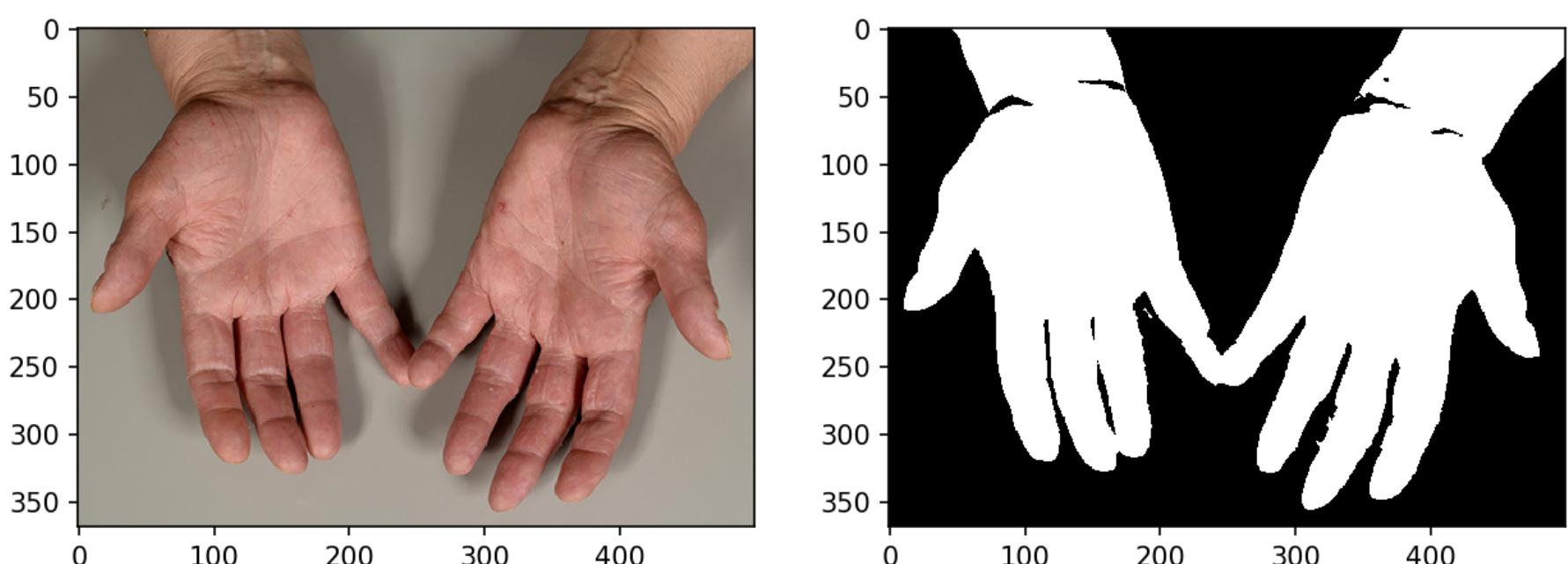
image_grey_back_hsv_hand = cv2.inRange(image_grey_back_hsv_blur, lower, upper)
image_grey_back_hsv_hand = cv2.bitwise_not(image_grey_back_hsv_hand)
image_grey_back_hsv_hand = cv2.erode(image_grey_back_hsv_hand, cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (3, 3)), iterations = 2)

## shadow
lower_shadow = np.array([100, 0, 90], dtype=np.uint8)
upper_shadow = np.array([179, 254, 254], dtype=np.uint8)
image_grey_back_hsv_hand_shadow = cv2.inRange(image_grey_back_hsv_blur, lower_shadow, upper_shadow)

# mix color- and shadow
image_grey_back_hsv_hand_and = cv2.bitwise_and(image_grey_back_hsv_hand, image_grey_back_hsv_hand_shadow)
```

```
In [7]: plt.subplot(1, 2, 1)
plt.imshow(image_grey_back)
plt.subplot(1, 2, 2)
plt.imshow(image_grey_back_hsv_hand_and, cmap='gray')
```

Out[7]: <matplotlib.image.AxesImage at 0x122afa5f8>

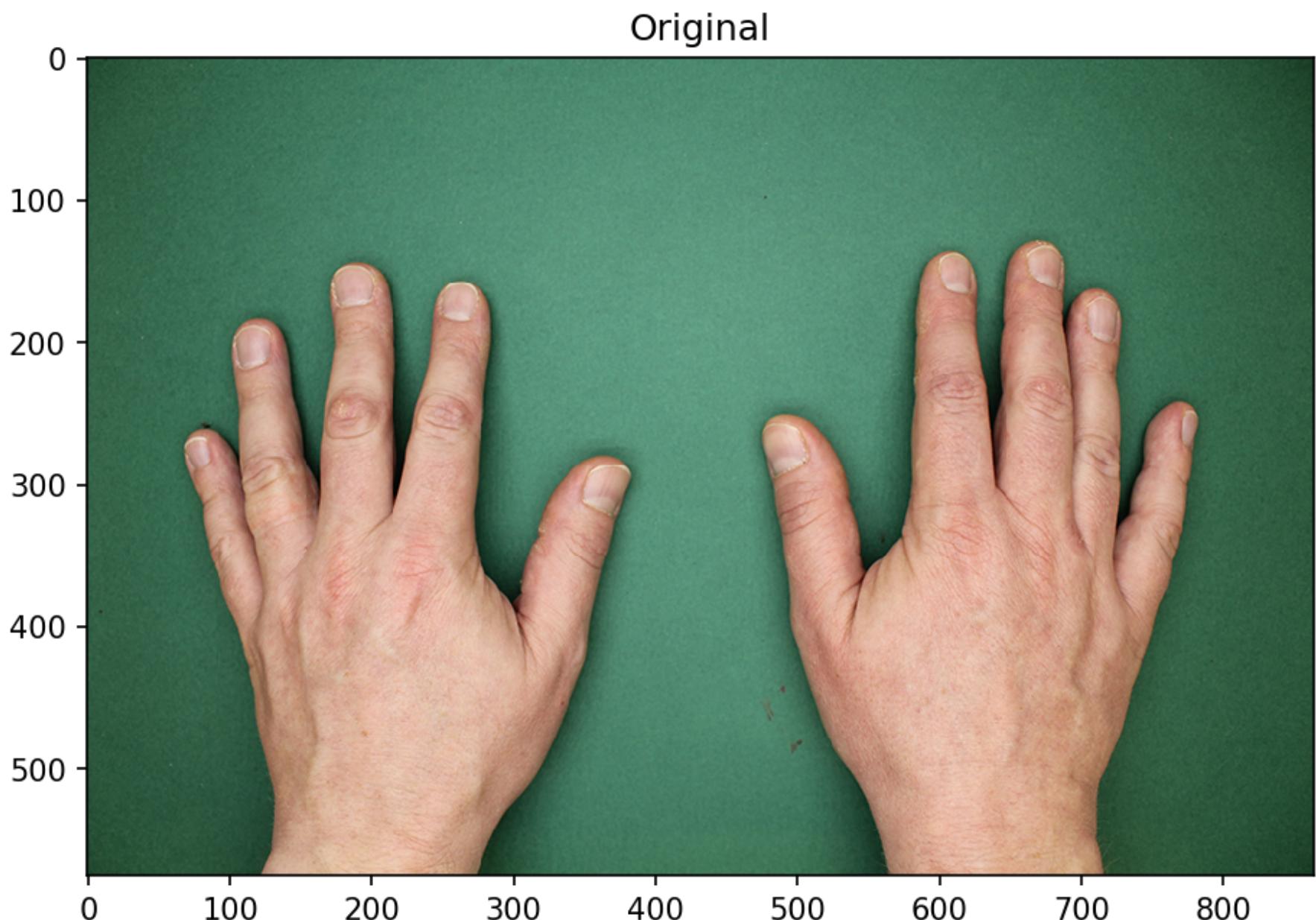


Aufgabe 2: Hintergrund Segmentierung

Im folgenden wird ein Bild mit grünem Hintergrund geladen. Führen Sie darauf die gleiche Segmentierung wie beim oberen Bild durch und stellen Sie diese dar.

```
In [8]: image_green_back = cv2.imread('../data/hand_green_small.png')
image_green_back = cv2.cvtColor(image_green_back, cv2.COLOR_BGR2RGB)
plt.title("Original")
plt.imshow(image_green_back)
```

```
Out[8]: <matplotlib.image.AxesImage at 0x1230f9898>
```



Aufgabe 2 mit HSV Farbraum

```
In [9]: image_green_back_hsv = cv2.cvtColor(image_green_back, cv2.COLOR_BGR2HSV)

# Adjust image a little bit (Anti-Aliasing)
image_green_back_hsv_blur = cv2.blur(image_green_back_hsv, (3,3))

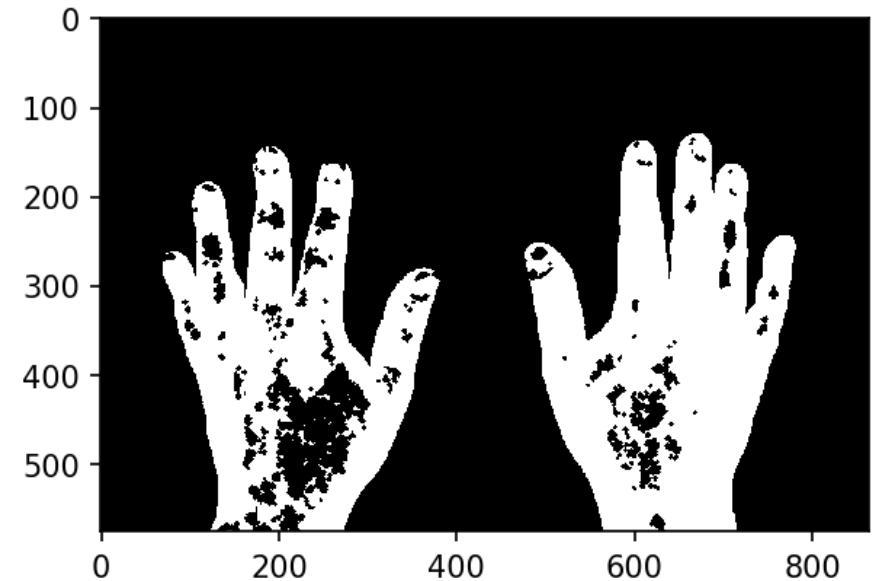
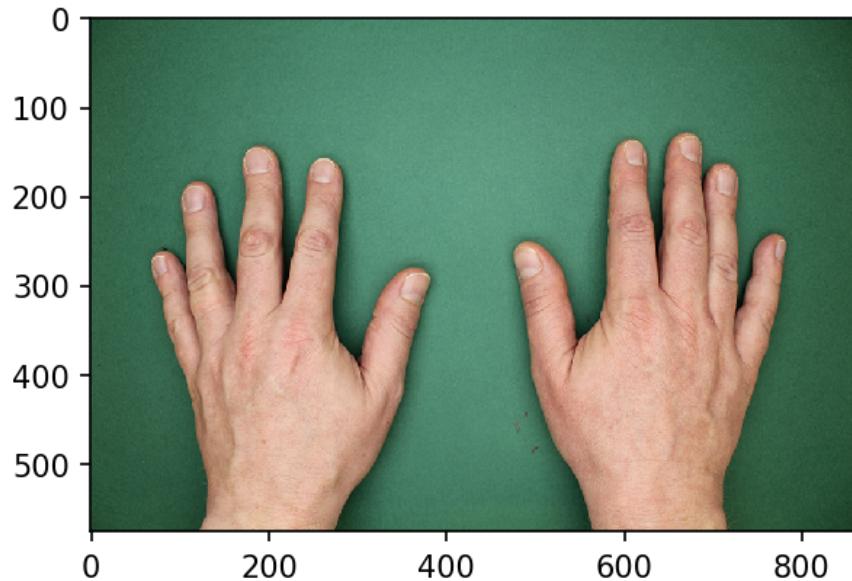
# Detect color- and shadow-areas of the hand
lower = np.array([30, 20, 0], dtype=np.uint8)
upper = np.array([110, 70, 254], dtype=np.uint8)
image_green_back_hsv_hand = cv2.inRange(image_green_back_hsv_blur, lower, upper)
image_green_back_hsv_hand = cv2.bitwise_not(image_green_back_hsv_hand)
image_green_back_hsv_hand = cv2.erode(image_green_back_hsv_hand, cv2.getStructuringElement(cv2.MORPH_ELLIPSE,
(3, 3)), iterations = 2)

## shadow
lower_shadow = np.array([100, 0, 90], dtype=np.uint8)
upper_shadow = np.array([179, 254, 254], dtype=np.uint8)
image_green_back_hsv_hand_shadow = cv2.inRange(image_green_back_hsv_blur, lower_shadow, upper_shadow)

# mix color- and shadow
image_green_back_hsv_hand_and = cv2.bitwise_and(image_green_back_hsv_hand,image_green_back_hsv_hand_shadow)

plt.subplot(1, 2, 1)
plt.imshow(image_green_back)
plt.subplot(1, 2, 2)
plt.imshow(image_green_back_hsv_hand_and, cmap='gray')
```

Out[9]: <matplotlib.image.AxesImage at 0x123cbf550>



Segmentieren Sie nun an Stelle der Haut den grünen Hintergrund. Können Sie damit eine bessere Unterscheidung zwischen Hand und Hintergrund erhalten? Stellen sie die beiden Bilder nebeneinander dar.

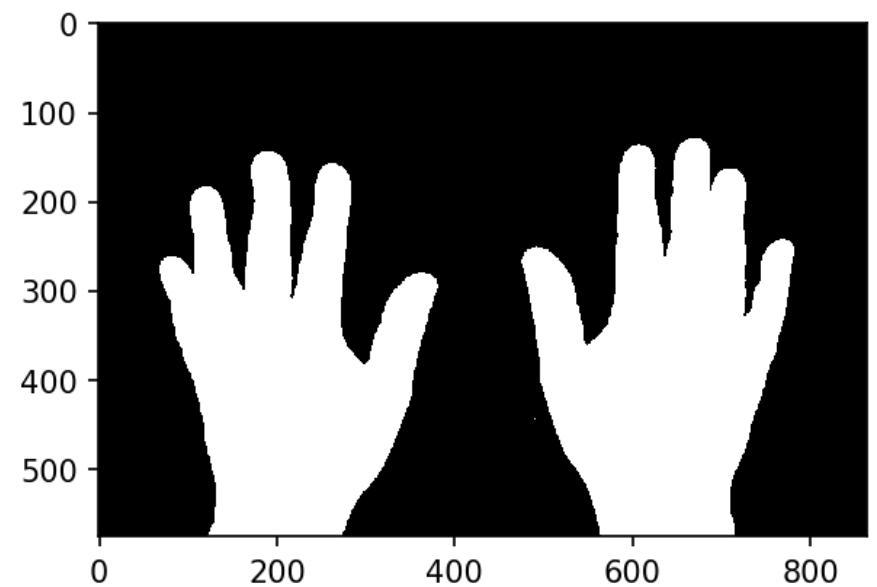
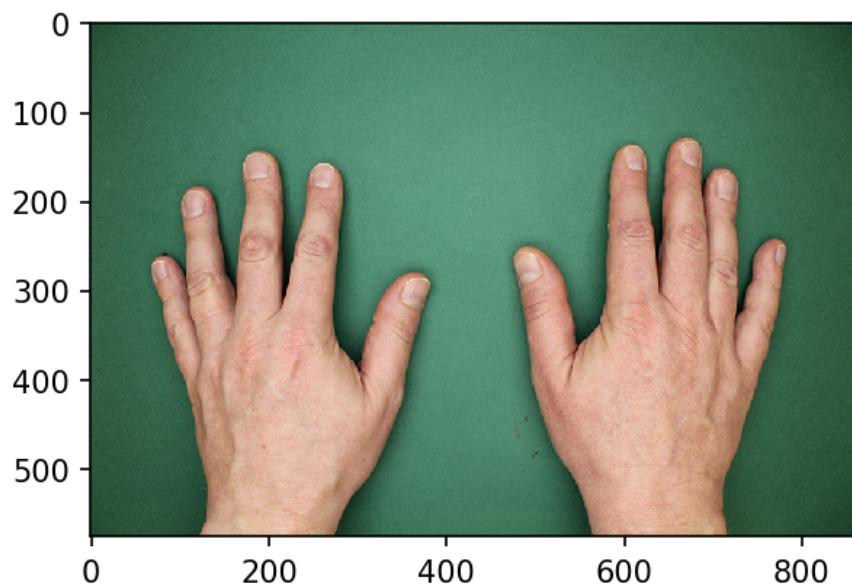
```
In [10]: image_green_back = cv2.imread('../data/hand_green_small.png')
image_green_back = cv2.cvtColor(image_green_back, cv2.COLOR_BGR2RGB)
image_green_back_hsv = cv2.cvtColor(image_green_back, cv2.COLOR_BGR2HSV)

# Adjust image a little bit
image_green_back_hsv_blur = cv2.blur(image_green_back_hsv, (3,3))

# Detect green background
lower_back = np.array([90, 20, 0], dtype=np.uint8)
upper_back = np.array([130, 254, 254], dtype=np.uint8)
image_green_back_only = cv2.inRange(image_green_back_hsv_blur, lower_back, upper_back)

plt.subplot(1, 2, 1)
plt.imshow(image_green_back)
plt.subplot(1, 2, 2)
plt.imshow(image_green_back_only, cmap='gray')
```

Out[10]: <matplotlib.image.AxesImage at 0x1242bd2e8>



Aufgabe 3: Bluescreen

Blue Screening wird in der Filmindustrie verwendet um Regionen in einem Bild, zum Beispiel ein blauer Hintergrund mit einem anderen Bild zu ersetzen.

Entwickeln sie ein Program, das 2 Bilder mit Hilfe dieser Technik aufeinander legt. Gehen Sie dazu wie folgt vor:

- Berechnen sie für jeden Pixel die Transparenz α aufgrund des Farbwertes von I_1
- Berechnen sie die Pixel des Ausgabebildes O , indem sie die beiden Bilder I_1 und I_2 mit Hilfe des α Werts übereinander legen. Eine mögliche Funktion dazu ist

$$O = \alpha * I_1 + (1 - \alpha) * I_2$$

- Dies wird als Alpha Blendung bezeichnet.

Ein Beispielbild mit blauem Hintergrund, sowie ein Beispiel Hintergrund werden im anschliessenden Code geladen. Allerdings haben die Bilder noch nicht die gleiche Grösse und müssen also zuerst noch skaliert werden. Sie können aber auch andere Bilder aus dem Internet verwenden.

Notizen Remo:

- In unserm Fall ist I_1 der Vordergrund (foreground) und I_2 der Hintergrund (background)
- Alpha Blendung => googling..

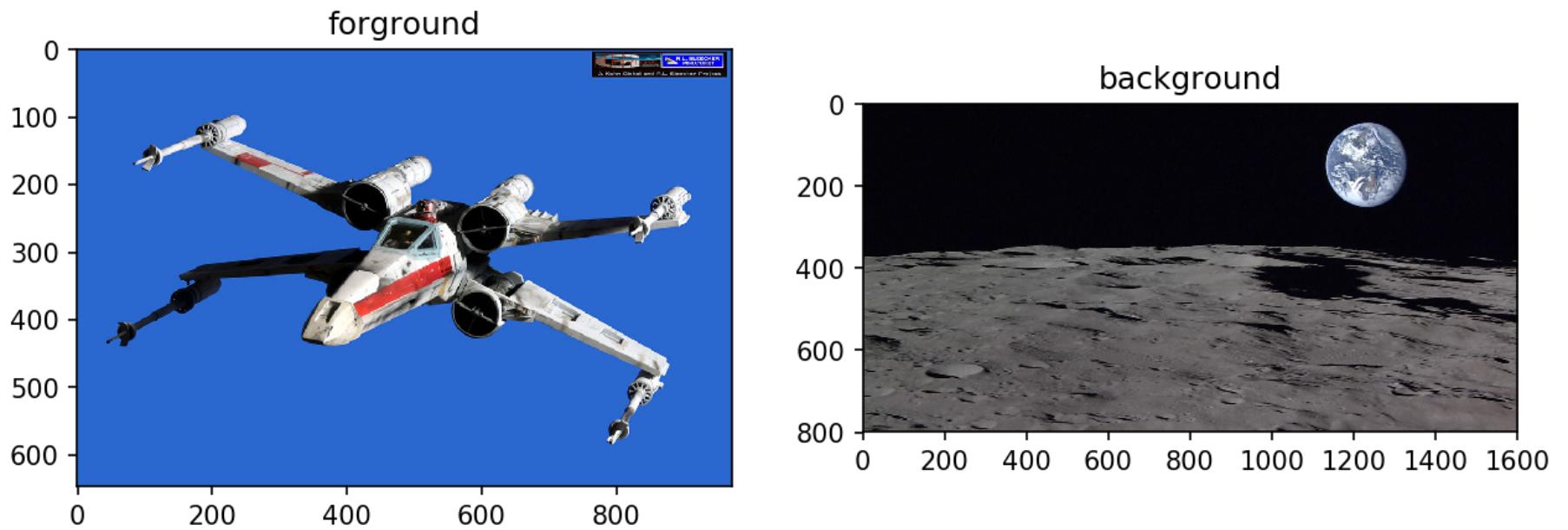
```
In [11]: # Foreground
foreground = cv2.imread('../data/2011-04-29_KG_RLB_RED_LEADER-003C_small.jpg')
foreground = cv2.cvtColor(foreground, cv2.COLOR_BGR2RGB)
# need for generating alpha-mask
foreground_hsv = cv2.cvtColor(foreground, cv2.COLOR_BGR2HSV)

# Background
background = cv2.imread('../data/landscape-moon.jpg')
background = cv2.cvtColor(background, cv2.COLOR_BGR2RGB)

plt.subplot(1, 2, 1)
plt.title("foreground")
plt.imshow(foreground)
plt.subplot(1, 2, 2)
plt.title("background")
plt.imshow(background)

print("foreground.shape = " + str(foreground.shape))
print("background.shape = " + str(background.shape))

foreground.shape = (648, 972, 3)
background.shape = (800, 1600, 3)
```



```
In [12]: # Size of Star Wars X-Wing Fighter image
row_size = len(foreground)
column_size = len(foreground[0])
# print(row_size) => 648
# print(column_size) => 972

# Calculate for each pixel the transparency  $\alpha$  based on the color value of the foreground
# values found from the internet
# For [H]ue (Farbton, Farbe)
minHue = 1
defHue = 11
maxHue = 21

# For [S]at (Sättigung, Reinheit)
minSat = 76
defSat = 203
maxSat = 254

# For [V]al (Intensität)
minVal = 76
defVal = 207
maxVal = 254

# For [H]ue
hueMask = np.logical_and(minHue <= foreground_hsv[:, :, 0], foreground_hsv[:, :, 0] <= maxHue)
# For [S]at
satMask = np.logical_and(minSat <= foreground_hsv[:, :, 1], foreground_hsv[:, :, 1] <= maxSat)
# For [V]al
valMask = np.logical_and(minVal <= foreground_hsv[:, :, 2], foreground_hsv[:, :, 2] <= maxVal)

# [H]ue AND [S]at
alphaMask = np.logical_and(hueMask, satMask)
#  $\alpha$  AND [V]al
alphaMask = np.logical_and(alphaMask, valMask)
alphaMask = np.invert(alphaMask)
alphaMask = alphaMask.astype(float)

# Adjust image a little bit (Anti-Aliasing)
alphaMask = cv2.blur(alphaMask, (3, 3))

# Remove logo
cv2.rectangle(alphaMask, (750, 0), (972, 50), (0), -1)

plt.imshow(alphaMask, cmap='gray')
```

Out[12]: <matplotlib.image.AxesImage at 0x125d519b0>



```
In [13]: # Generate a new array for the final image
finalImage = background.copy()

# set position of the X-Wing fighter in the background image
offset_x = 400
offset_y = 150

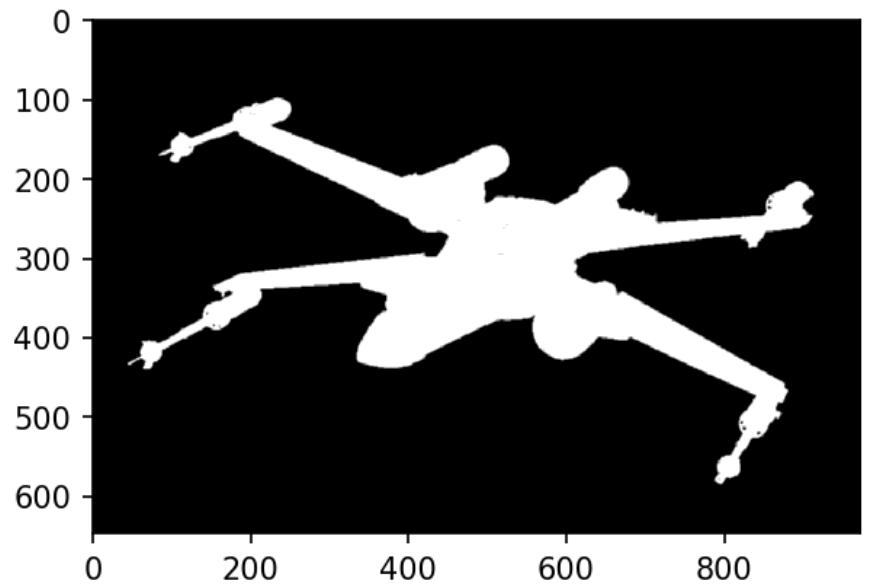
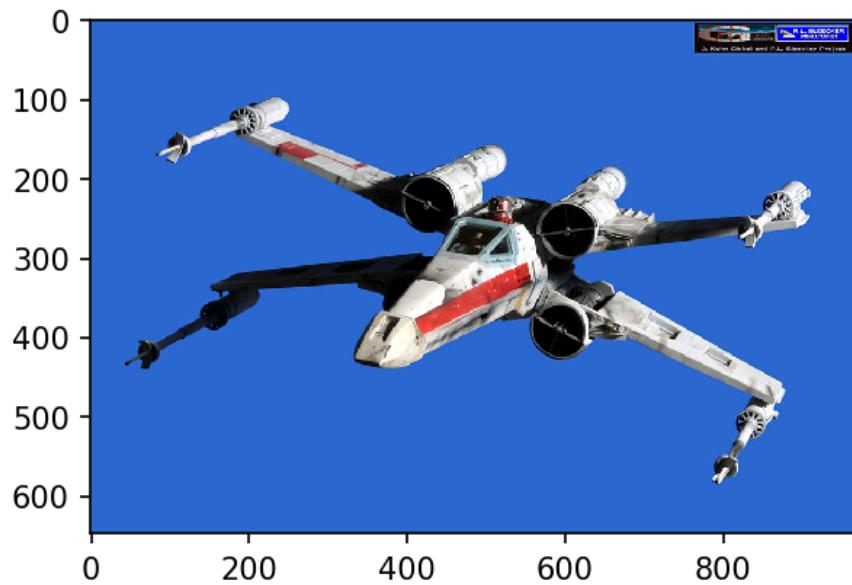
# Merge both images, together with the previous generated alpha-mask
for row in range(0, row_size):
    for column in range(0, column_size):
        curr_background_pixel = finalImage[row + offset_y, column + offset_x]
        curr_foreground_pixel = foreground[row, column]
        curr_alpha_value = alphaMask[row, column]

        #  $O = foreground * \alpha + background * (1 - \alpha)$ 
        # one pixel = 3 values!
        for pixel_value in range(0, 3):
            curr_background_pixel[pixel_value] = curr_foreground_pixel[pixel_value] * curr_alpha_value + curr_background_pixel[pixel_value] * (1 - curr_alpha_value)

    finalImage[row + offset_y, column + offset_x] = curr_background_pixel

# Plotting
plt.subplot(1, 2, 1)
plt.imshow(foreground/255)
plt.subplot(1, 2, 2)
plt.imshow(alphaMask, cmap='gray')
```

Out[13]: <matplotlib.image.AxesImage at 0x1274be438>



```
In [14]: # Plotting final image
plt.imshow(finalImage/255)
```

Out[14]: <matplotlib.image.AxesImage at 0x12580ba58>

