Bartosz Klimza

Sprawozdanie 2 - Przetwarzanie i analiza obrazów binarnych

Celem laboratorium jest podstawowe przetwarzanie i analiza obrazów binarnych. W tym celu zostaną wyekstrahowane trzema metodami szkielety na obrazach przedstawiających kości, a następnie wygenerowane linie aproksymujące te szkielety.

Wykorzystany kod znajduje się w pliku skeletons.py, a skrypt uruchomieniowy w pliku skeletons.sh.

Na początek ładuję biblioteki:

```python
import matplotlib.pyplot as plt
import os
from PIL import Image
from skimage import io
from skimage.morphology import skeletonize, thin
from skimage.color import rgb2gray
from skimage.transform import probabilistic_hough_line
```

Następnie zmieniam kolory na obrazach: czerwony, zielony i niebieski stają się białe, a tło z szarego na czarne. Wykorzystuję do tego następującą funkcję:

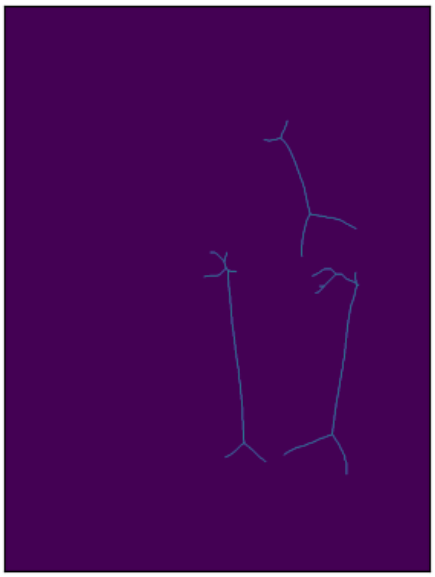
```python
def change_image_to_bw(path):
    img = Image.open(path)
    width = img.size[0]
    height = img.size[1]
    for i in range(0, width):
        for j in range(0, height):
            data = img.getpixel((i, j))
            if data[0] == 255 and data[1] == 0 and data[2] == 0:
                img.putpixel((i, j), (255, 255, 255))
            if data[0] == 0 and data[1] == 255 and data[2] == 0:
                img.putpixel((i, j), (255, 255, 255))
            if data[0] == 0 and data[1] == 0 and data[2] == 255:
                img.putpixel((i, j), (255, 255, 255))
            if data[0] == 32 and data[1] == 32 and data[2] == 32:
                img.putpixel((i, j), (0, 0, 0))
    return img
```

Wygeneruję szkielety na obrazach za pomocą trzech metod: standardowej, lee oraz thin, która redukuje zbędne odnogi szkieletu, zatem spodziewam się po niej najlepszego wyniku. Wykorzystam do tego poniższy fragment kodu:

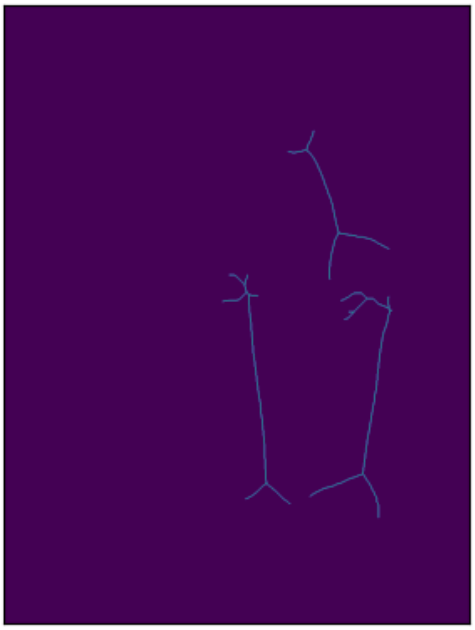
```
for image in os.listdir(data):
    print(image)
    img = change_image_to_bw(f"{data}{image}")
    img.save(f"bw_{image}")
    img = rgb2gray(io.imread(f"bw_{image}"))

    skeletonized_img = skeletonize(img)
    plt.imshow(img, alpha=1)
    plt.imshow(skeletonized_img, alpha=0.5)
    plt.xticks([])
    plt.yticks([])
    plt.savefig(f"./results/skeletons/{image}")

    # Metoda lee
    skeletonized_lee_img = skeletonize(img, method='lee')
    plt.imshow(img, alpha=1)
    plt.imshow(skeletonized_img, alpha=0.5)
    plt.xticks([])
    plt.yticks([])
    plt.savefig(f"./results/skeletons/lee_{image}")

    # Metoda thin
    skeletonized_thin_img = thin(img)
    plt.imshow(img, alpha=1)
    plt.imshow(skeletonized_thin_img, alpha=0.5)
    plt.xticks([])
    plt.yticks([])
    plt.savefig(f"./results/skeletons/thin_{image}")
```

Przykładowe wyniki dla obrazu „img_0007_l”:

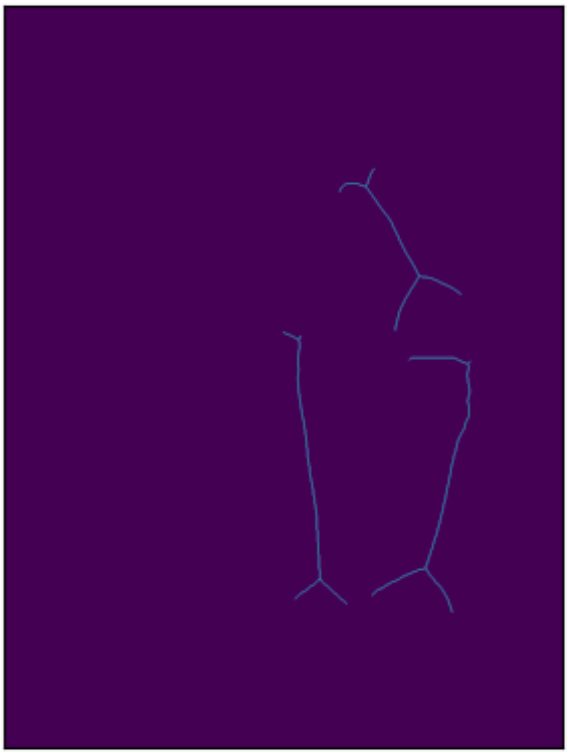
Metoda standardowa:



Metoda lee:



Metoda thin:



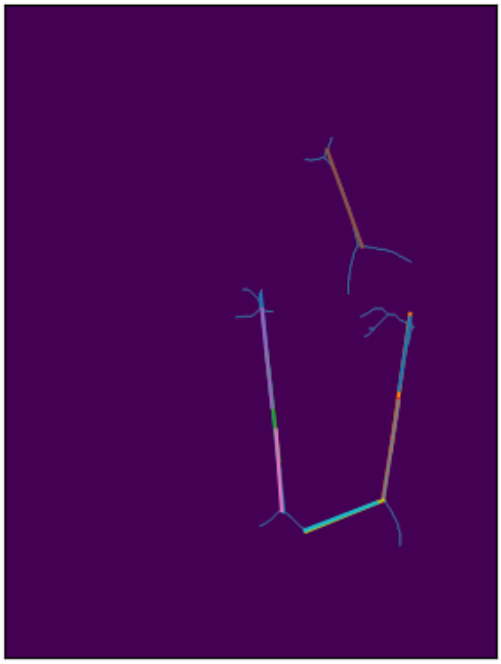
Można zauważyć, że metody standardowa oraz lee radzą sobie dość podobnie. Metoda thin radzi sobie najlepiej upraszczając szkielet.

Kolejnym krokiem jest aproksymacja szkieletów za pomocą linii. Do ich wygenerowania stosuję funkcję probabilistic_hough_line dostępną w bibliotece skimage:

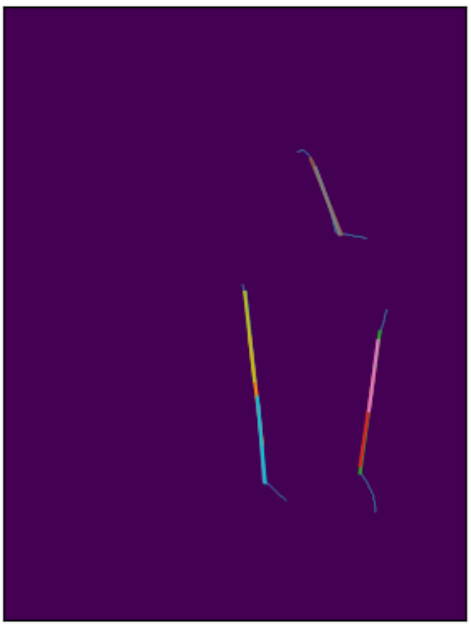
```python
# Linie aproksymujące szkielety tworzone metodą klasyczną
angle = probabilistic_hough_line(skeletonized_img, threshold=15, line_length=110, line_gap=50)
plt.imshow(skeletonized_img)
plt.xticks([])
plt.yticks([])
for line in angle:
    p0, p1 = line
    plt.plot((p0[0], p1[0]), (p0[1], p1[1]))
plt.savefig(f"./results/lines/{image}")
plt.clf()
with open("./results/hallux_angles.txt", "a") as f:
    f.write(f"{image} - {angle}\n")

# Linie aproksymujące szkielety tworzone metodą lee
angle_lee = probabilistic_hough_line(skeletonized_lee_img, threshold=15, line_length=110, line_gap=50)
plt.imshow(skeletonized_lee_img)
plt.xticks([])
plt.yticks([])
for line in angle_lee:
    p0, p1 = line
    plt.plot((p0[0], p1[0]), (p0[1], p1[1]))
plt.savefig(f"./results/lines/lee_{image}")
plt.clf()
with open("./results/hallux_angles.txt", "a") as f:
    f.write(f"lee_{image} - {angle}\n")

# Linie aproksymujące szkielety tworzone metodą thin
angle_thinned = probabilistic_hough_line(skeletonized_thin_img, threshold=15, line_length=110, line_gap=50)
plt.imshow(skeletonized_thin_img)
plt.xticks([])
plt.yticks([])
for line in angle_thinned:
    p0, p1 = line
    plt.plot((p0[0], p1[0]), (p0[1], p1[1]))
plt.savefig(f"./results/lines/thin_{image}")
plt.clf()
with open("./results/hallux_angles.txt", "a") as f:
    f.write(f"thin_{image} - {angle}\n")
```

Przykładowe wyniki dla obrazu „img_0007_l”:

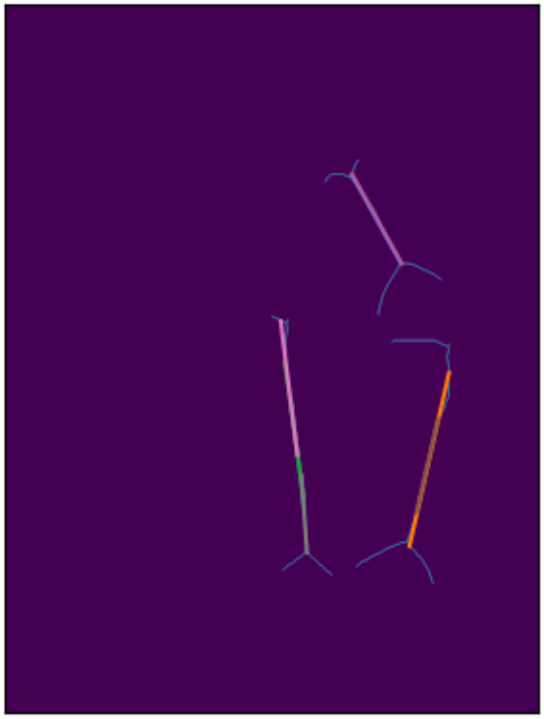
Metoda standardowa:



Metoda lee:



Metoda thin:



Parametry wyznaczonych linii znajdują się w pliku hallux_angles.txt.