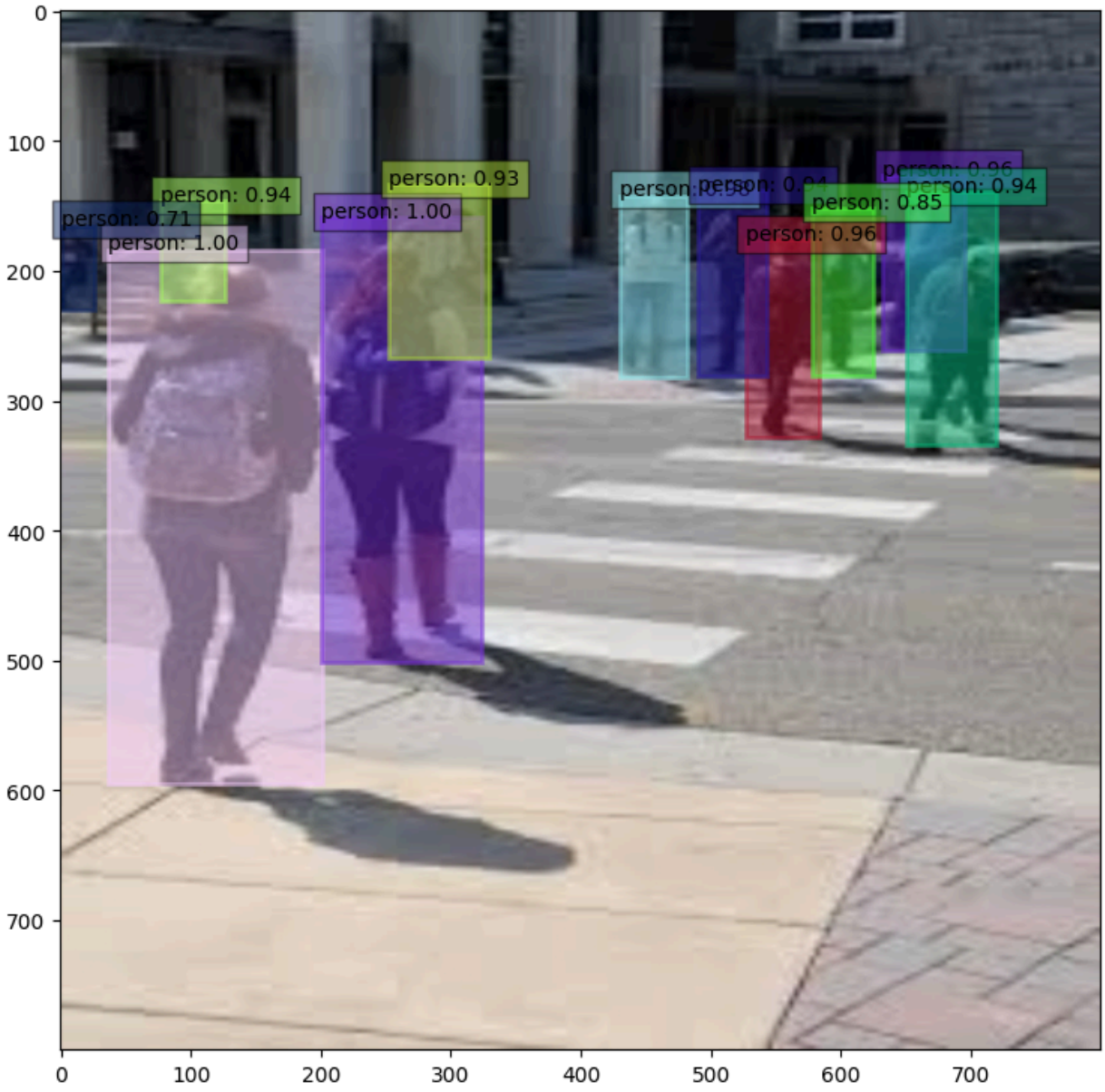
Image Segmentation

In [ ]:

```
import torch
import torchvision
from PIL import Image
import matplotlib.pyplot as plt
import torchvision.transforms as T
import random

model = torchvision.models.detection.fasterrcnn_resnet50_fpn(pretrained=True)
model.eval()

img_path = 'Picture1.jpg'
img = Image.open(img_path).convert("RGB")

transform = T.Compose([
    T.Resize((800, 800)),
    T.ToTensor()
])
img = transform(img)

img = img.unsqueeze(0)

with torch.no_grad():
    prediction = model(img)

confidence_threshold = 0.7

CATEGORY_NAMES = [
    '__background__', 'person', 'bicycle', 'car', 'motorcycle', 'airplane', 'bus', 'tr
    'boat', 'traffic light', 'fire hydrant', 'stop sign', 'parking meter', 'bench', 'b
    'dog', 'horse', 'sheep', 'cow', 'elephant', 'bear', 'zebra', 'giraffe', 'backpack'
    'handbag', 'tie', 'suitcase', 'frisbee', 'skis', 'snowboard', 'sports ball', 'kite
    'baseball glove', 'skateboard', 'surfboard', 'tennis racket', 'bottle', 'wine glas
    'knife', 'spoon', 'bowl', 'banana', 'apple', 'sandwich', 'orange', 'broccoli', 'ca
    'pizza', 'donut', 'cake', 'chair', 'couch', 'potted plant', 'bed', 'dining table',
    'laptop', 'mouse', 'remote', 'keyboard', 'cell phone', 'microwave', 'oven', 'toast
    'refrigerator', 'book', 'clock', 'vase', 'scissors', 'teddy bear', 'hair drier', '
]

target_classes = ['person', 'car', 'sandwich', 'cup', 'fork', 'knife', 'spoon', 'bowl'
target_class_indices = [CATEGORY_NAMES.index(cls) for cls in target_classes]

labels = prediction[0]['labels']
scores = prediction[0]['scores']
boxes = prediction[0]['boxes']

high_conf_indices = [i for i, (label, score) in enumerate(zip(labels, scores)) if scor

filtered_boxes = boxes[high_conf_indices]
filtered_labels = labels[high_conf_indices]
filtered_scores = scores[high_conf_indices]

def plot_image(image, boxes, labels, scores):
    fig, ax = plt.subplots(1, figsize=(12, 9))
    ax.imshow(image)
    for box, label, score in zip(boxes, labels, scores):
        box = box.cpu().numpy()
```

```
        color = [random.random() for _ in range(3)]
        rect = plt.Rectangle(
            (box[0], box[1]),
            box[2] - box[0],
            box[3] - box[1],
            fill=True,
            color=color,
            alpha=0.5,
            edgecolor=color,
            linewidth=2
        )
        ax.add_patch(rect)
        ax.text(box[0], box[1], f'{CATEGORY_NAMES[label]}: {score:.2f}', bbox=dict(fac
    plt.show()

img_np = img.squeeze().permute(1, 2, 0).cpu().numpy()

plot_image(img_np, filtered_boxes, filtered_labels, filtered_scores)
```

```
c:\Users\rowen\AppData\Local\Programs\Python\Python311\Lib\site-packages\torchvision
\models\_utils.py:208: UserWarning: The parameter 'pretrained' is deprecated since 0.
13 and may be removed in the future, please use 'weights' instead.
  warnings.warn(
c:\Users\rowen\AppData\Local\Programs\Python\Python311\Lib\site-packages\torchvision
\models\_utils.py:223: UserWarning: Arguments other than a weight enum or `None` for
'weights' are deprecated since 0.13 and may be removed in the future. The current beh
avior is equivalent to passing `weights=FasterRCNN_ResNet50_FPN_Weights.COCO_V1`. You
can also use `weights=FasterRCNN_ResNet50_FPN_Weights.DEFAULT` to get the most up-to-
date weights.
  warnings.warn(msg)
C:\Users\rowen\AppData\Local\Temp\ipykernel_12620\3609208464.py:58: UserWarning: Sett
ing the 'color' property will override the edgecolor or facecolor properties.
  rect = plt.Rectangle(
```



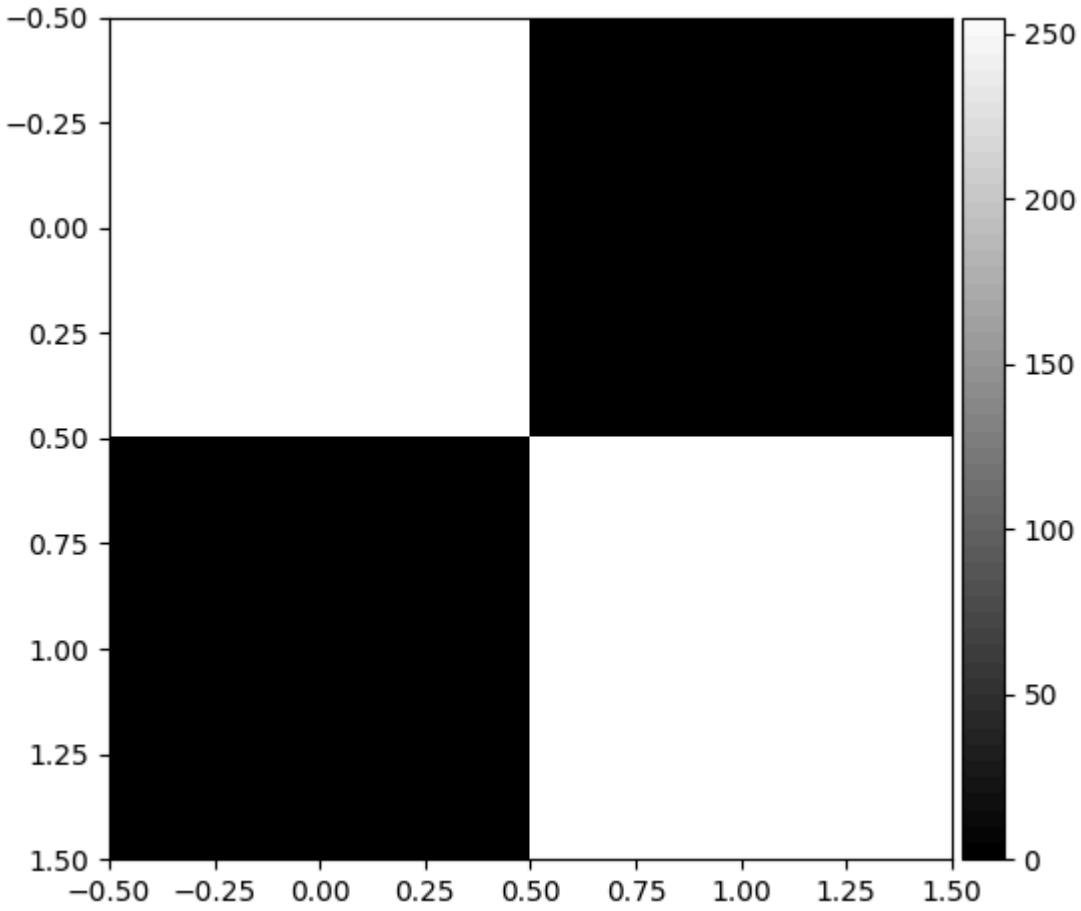
Representation

In [ ]:
```
import numpy as np
import matplotlib.pyplot as plt
from skimage.color import rgb2gray
from skimage.io import imshow
from skimage import img_as_ubyte
import cv2
```

In [ ]:
```
array = np.array([[255,0], [0,255]])

imshow(array, cmap='gray')
```

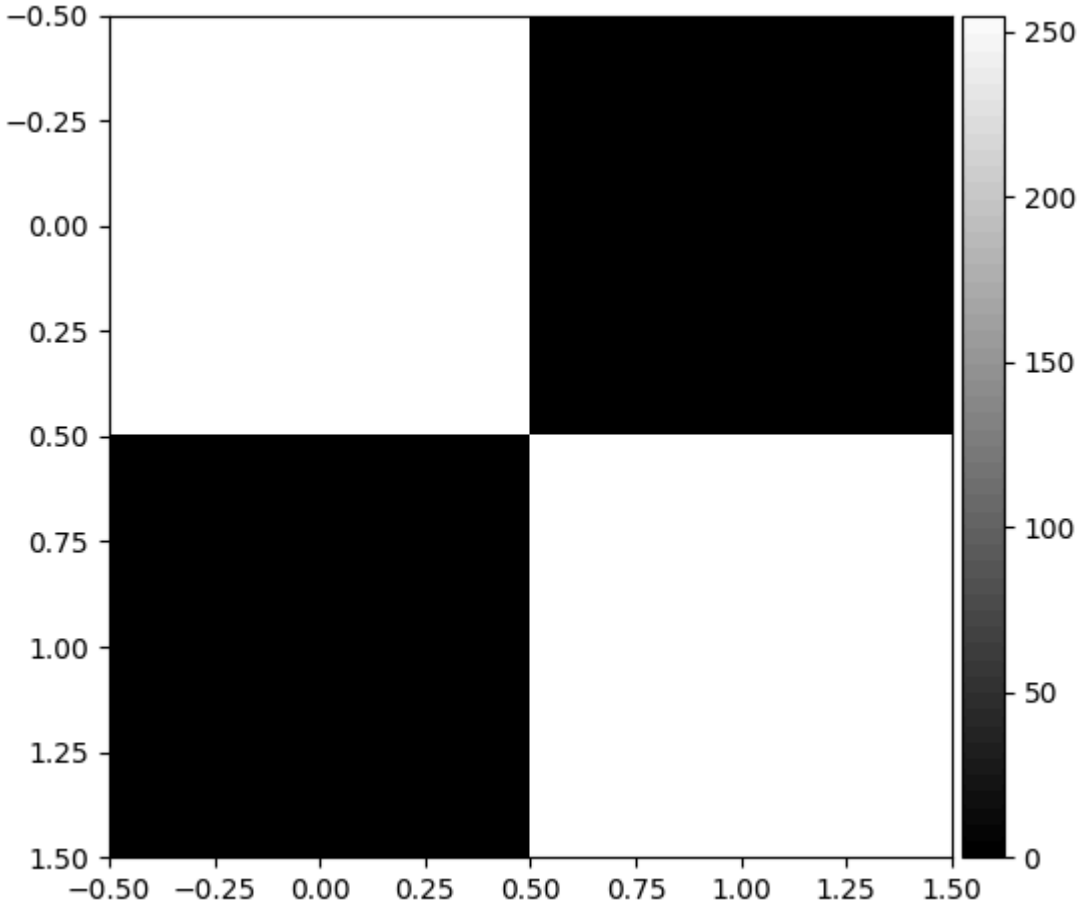
```
c:\Users\rowen\AppData\Local\Programs\Python\Python311\Lib\site-packages\skimage\io\_
plugins\matplotlib_plugin.py:158: UserWarning: Low image data range; displaying image
with stretched contrast.
  lo, hi, cmap = _get_display_range(image)
```

Out[ ]:
```
<matplotlib.image.AxesImage at 0x10df826de10>
```

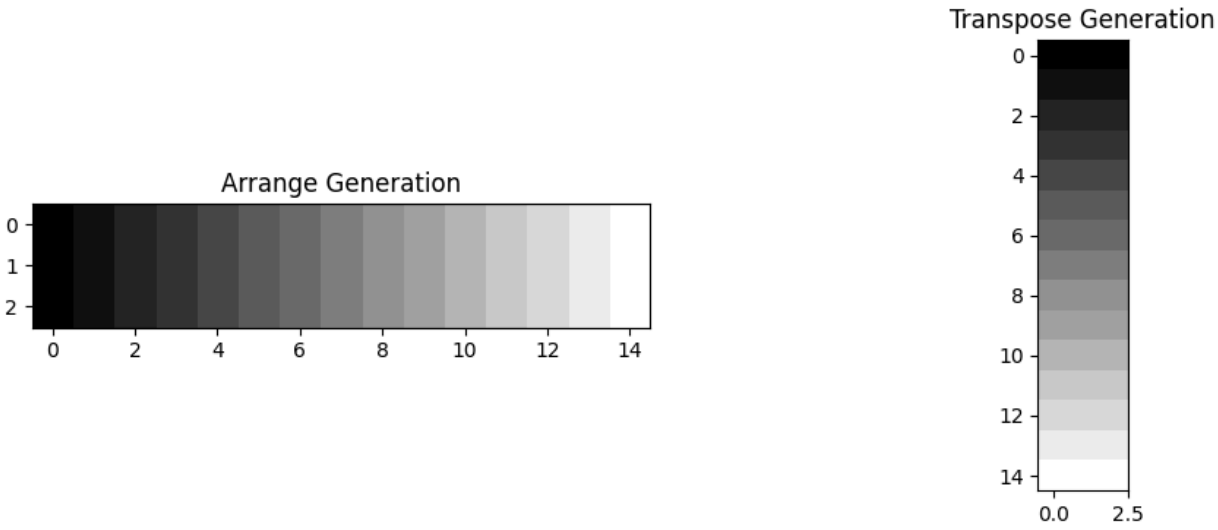


In [ ]:
```
array = np.array([[255,0], [0,255]])

imshow(array, cmap='gray')
```

Out[ ]:
```
<matplotlib.image.AxesImage at 0x10dfee30ed0>
```

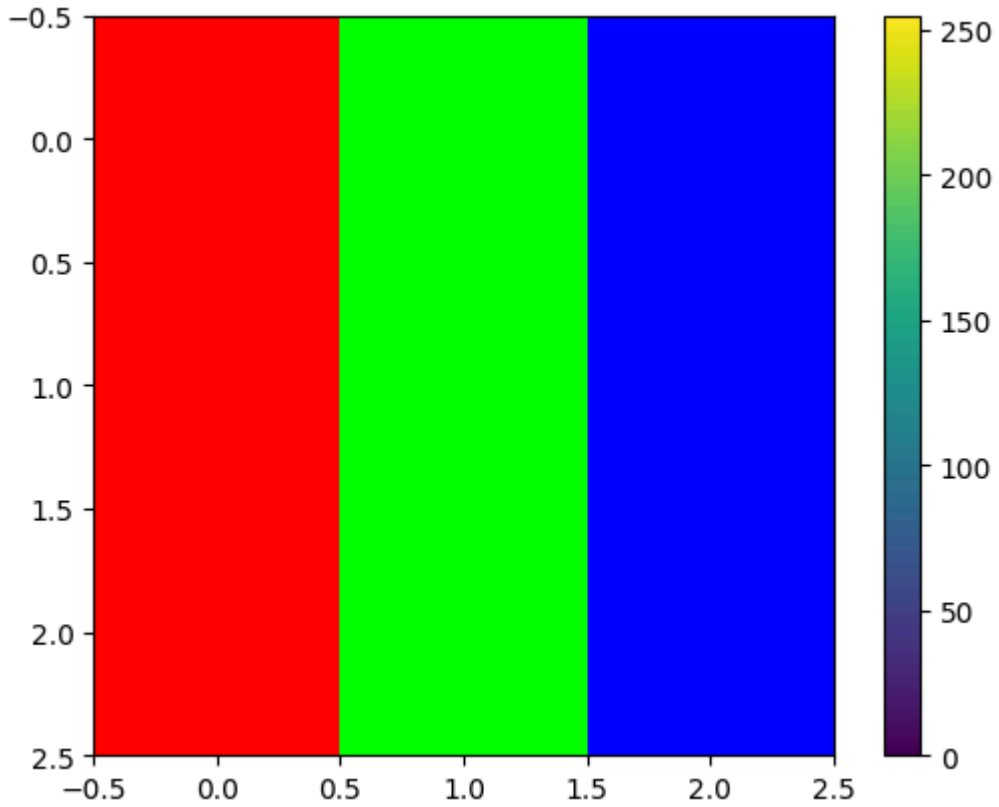


In [ ]:
```
array_spectrum = np.array([np.arange(0, 255, 17), np.arange(0, 255, 17)])
array_spectrum = np.append(array_spectrum, [np.arange(0, 255, 17)], axis=0)

fig, ax = plt.subplots(1, 2, figsize=(12, 4))
ax[0].imshow(array_spectrum, cmap='gray')
ax[0].set_title('Arrange Generation')
ax[1].imshow(array_spectrum.T, cmap='gray')
ax[1].set_title('Transpose Generation')
plt.show()
```



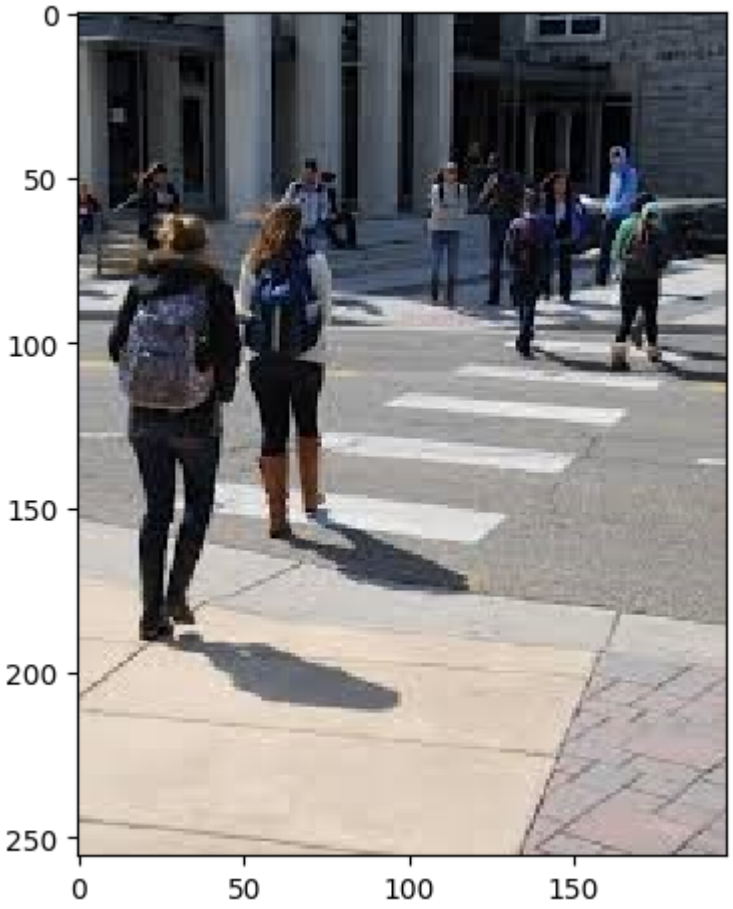
In [ ]:
```
array_colors = np.array([[[255, 0, 0], [0, 255, 0], [0, 0, 255]],
                         [[255, 0, 0], [0, 255, 0], [0, 0, 255]],
```

```
                        [[255, 0, 0], [0, 255, 0], [0, 0, 255]]], dtype=np.uint8)

# Display the image
plt.imshow(array_colors)
plt.colorbar()
plt.show()
```



In [ ]:
```
draft = plt.imread('Picture1.jpg')
imshow(draft)
```

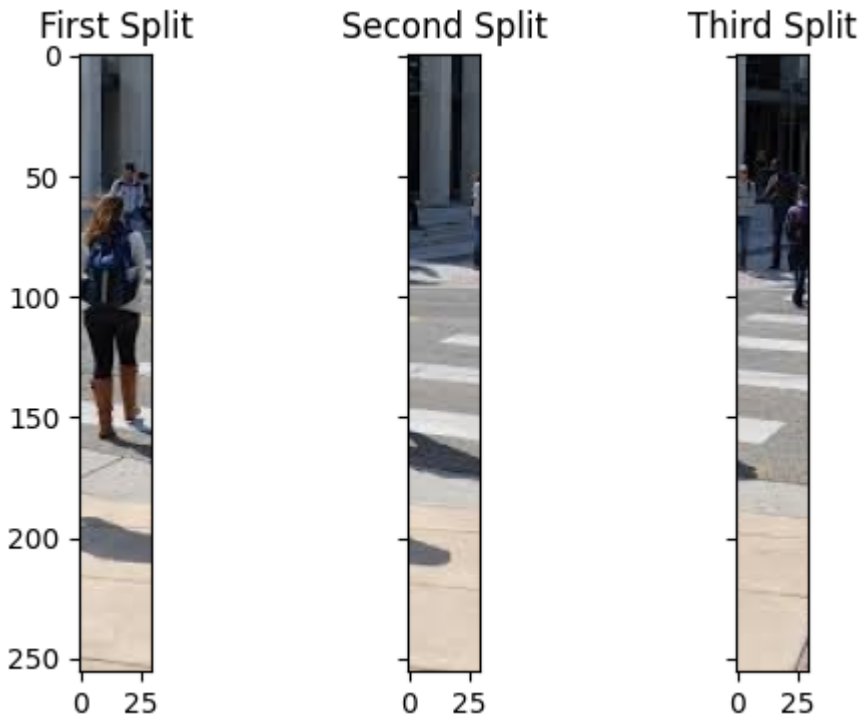
Out[ ]:
```
<matplotlib.image.AxesImage at 0x10d9c3d6e10>
```



In [ ]:
```
draft.shape
```

Out[ ]:
```
(256, 197, 3)
```

In [ ]:
```
fig, ax = plt.subplots(1, 3,
                       figsize=(6, 4),
                       sharey=True)

ax[0].imshow(draft[:, 50:80])
ax[0].set_title('First Split')

ax[1].imshow(draft[:, 80:110])
ax[1].set_title('Second Split')

ax[2].imshow(draft[:, 110:140])
ax[2].set_title('Third Split')

plt.show()
```



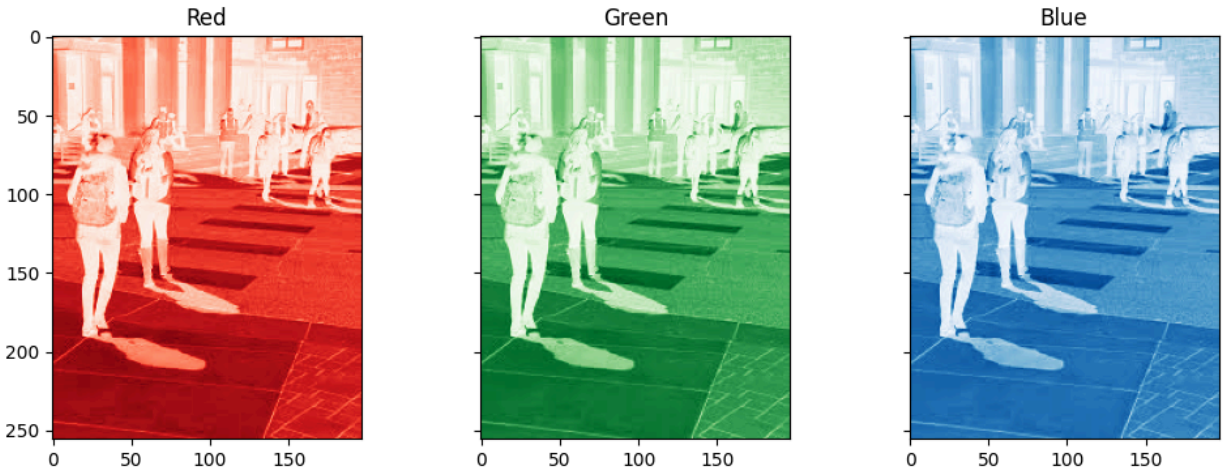
In [ ]:
```
imshow(draft[50:200, 80:200])
```

Out[ ]:
```
<matplotlib.image.AxesImage at 0x10d9c364990>
```

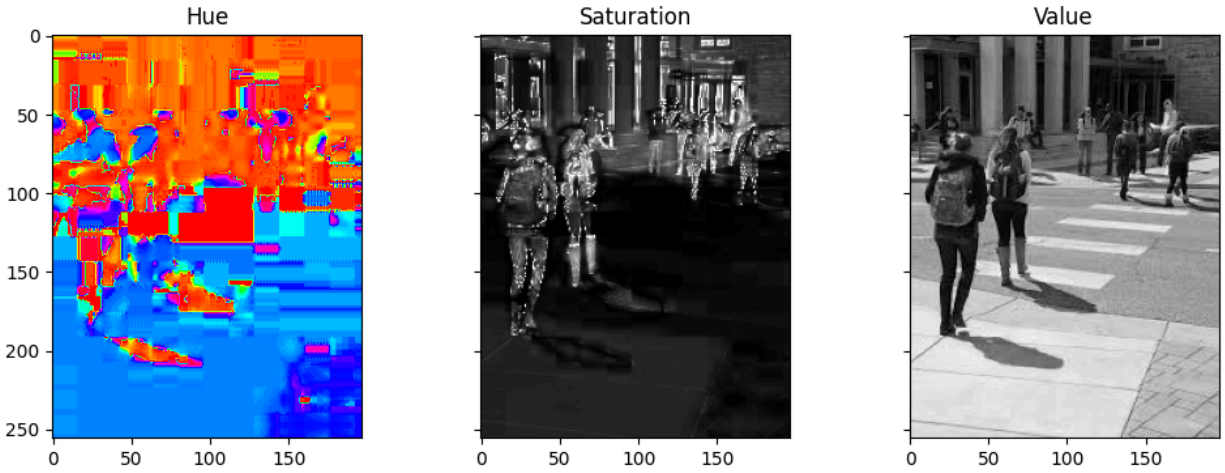
In [ ]:
```
fig, ax = plt.subplots(1, 3, figsize=(12, 4), sharey=True)
```

```
ax[0].imshow(draft[:, :, 0], cmap='Reds')
ax[0].set_title('Red')

ax[1].imshow(draft[:, :, 1], cmap='Greens')
ax[1].set_title('Green')

ax[2].imshow(draft[:, :, 2], cmap='Blues')
ax[2].set_title('Blue')
```

Out[ ]:
```
Text(0.5, 1.0, 'Blue')
```

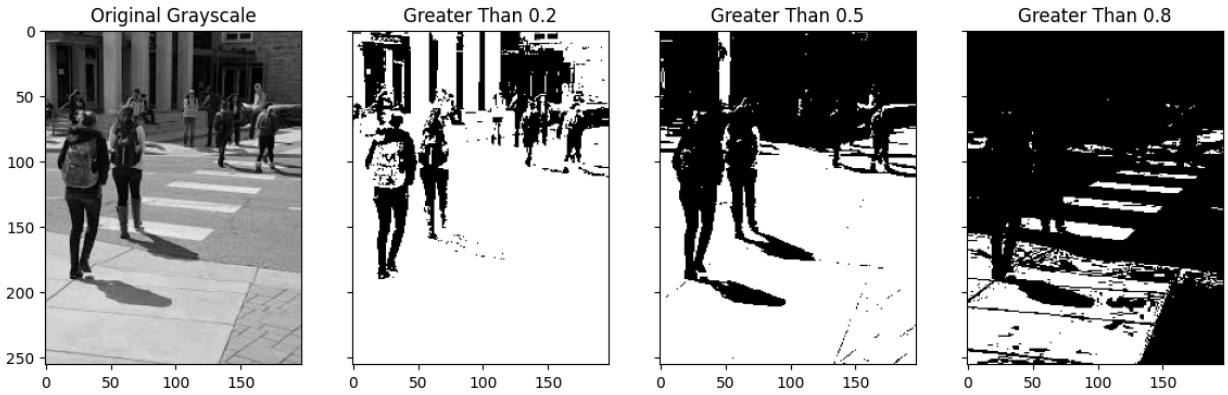


In [ ]:
```
draft_hsv = cv2.cvtColor(draft, cv2.COLOR_BGR2HSV)
fig, ax = plt.subplots(1, 3, figsize=(12,4), sharey=True)
ax[0].imshow(draft_hsv[:, :, 0], cmap='hsv')
ax[0].set_title('Hue')
ax[1].imshow(draft_hsv[:, :, 1], cmap='gray')
ax[1].set_title('Saturation')
ax[2].imshow(draft_hsv[:, :, 2], cmap='gray')
ax[2].set_title('Value')
plt.show()
```



In [ ]:
```
draft_gray = cv2.cvtColor(draft, cv2.COLOR_BGR2GRAY)
fig, ax = plt.subplots(1, 4, figsize=(14, 4), sharey=True)
ax[0].imshow(draft_gray, cmap='gray')
ax[0].set_title('Original Grayscale')
ax[1].imshow(draft_gray > 50, cmap='gray')
ax[1].set_title('Greater Than 0.2')
ax[2].imshow(draft_gray > 125, cmap='gray')
```

```
ax[2].set_title('Greater Than 0.5')
ax[3].imshow(draft_gray > 200, cmap='gray')
ax[3].set_title('Greater Than 0.8')
plt.show()
```



Description

In [ ]:
```
from skimage import io, img_as_ubyte
import matplotlib.pyplot as plt
from skimage.color import label2rgb, rgb2gray
import numpy as np
from skimage.filters import threshold_otsu
from skimage.segmentation import clear_border
from skimage import measure
import pandas as pd
```

In [ ]:
```
image = img_as_ubyte(io.imread('Picture1.jpg', as_gray=True))

plt.imshow(image, cmap='gray')
plt.axis('off')
plt.show()
```

In [ ]:
```
threshold = threshold_otsu(image)
thresholded_img = image > threshold
plt.imshow(thresholded_img)
plt.axis('off')
```

Out[ ]:
```
(-0.5, 196.5, 255.5, -0.5)
```

In [ ]:
```
removed = clear_border(thresholded_img)
plt.imshow(removed)
plt.axis('off')
```

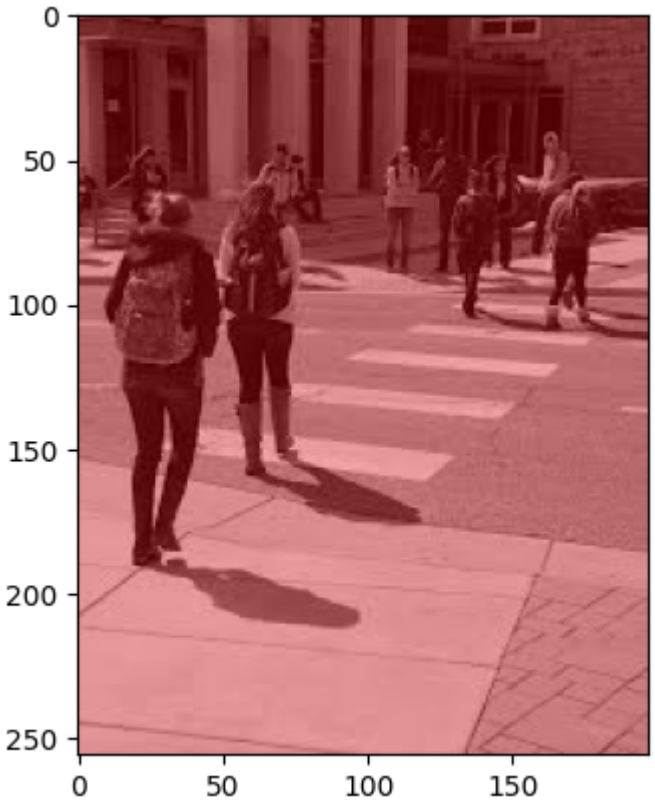
Out[ ]:
```
(-0.5, 196.5, 255.5, -0.5)
```

In [ ]:
```
label_image = measure.label(removed, connectivity=image.ndim)
plt.imshow(label_image)
plt.axis('off')
```

Out[ ]:
```
(-0.5, 196.5, 255.5, -0.5)
```

In [ ]:
```
image_label_overlay = label2rgb(label_image, image=image, bg_label=1, colors=['red'])

plt.imshow(image_label_overlay)
plt.show()
```

```
plt.imsave('labeled_cast.jpg', image_label_overlay)
```



In [ ]:
```
all_props = measure.regionprops(label_image,image)
```

In [ ]:
```
for prop in all_props:
    print("Label : {}, Area : {}".format(prop.label, prop.area))
```

```
Label : 1, Area : 1.0
Label : 2, Area : 1.0
Label : 3, Area : 1.0
Label : 4, Area : 2.0
Label : 5, Area : 2.0
Label : 6, Area : 5.0
Label : 7, Area : 1.0
Label : 8, Area : 2.0
Label : 9, Area : 2.0
Label : 10, Area : 3.0
Label : 11, Area : 6.0
Label : 12, Area : 1.0
Label : 13, Area : 3.0
Label : 14, Area : 2.0
Label : 15, Area : 3.0
Label : 16, Area : 1.0
Label : 17, Area : 4.0
Label : 18, Area : 5.0
Label : 19, Area : 55.0
Label : 20, Area : 3.0
Label : 21, Area : 38.0
Label : 22, Area : 1.0
Label : 23, Area : 1.0
Label : 24, Area : 21.0
Label : 25, Area : 1.0
Label : 26, Area : 1.0
Label : 27, Area : 1.0
Label : 28, Area : 1.0
Label : 29, Area : 1.0
Label : 30, Area : 9.0
Label : 31, Area : 3.0
Label : 32, Area : 1.0
Label : 33, Area : 1.0
Label : 34, Area : 3.0
Label : 35, Area : 3.0
Label : 36, Area : 1.0
Label : 37, Area : 29.0
Label : 38, Area : 11.0
Label : 39, Area : 1.0
Label : 40, Area : 1.0
Label : 41, Area : 6.0
Label : 42, Area : 1.0
Label : 43, Area : 5.0
Label : 44, Area : 1.0
Label : 45, Area : 1.0
Label : 46, Area : 10.0
Label : 47, Area : 77.0
Label : 48, Area : 10.0
Label : 49, Area : 2.0
Label : 50, Area : 2.0
Label : 51, Area : 11.0
Label : 52, Area : 74.0
Label : 53, Area : 9.0
Label : 54, Area : 1.0
Label : 55, Area : 1.0
Label : 56, Area : 1.0
Label : 57, Area : 3.0
Label : 58, Area : 94.0
Label : 59, Area : 2.0
Label : 60, Area : 1.0
```

```
Label : 61, Area : 9.0
Label : 62, Area : 9.0
Label : 63, Area : 4.0
Label : 64, Area : 293.0
Label : 65, Area : 5.0
Label : 66, Area : 1.0
Label : 67, Area : 4.0
Label : 68, Area : 1.0
Label : 69, Area : 1.0
Label : 70, Area : 1.0
Label : 71, Area : 2.0
Label : 72, Area : 3.0
Label : 73, Area : 1.0
Label : 74, Area : 1.0
Label : 75, Area : 90.0
Label : 76, Area : 7.0
Label : 77, Area : 1.0
Label : 78, Area : 1.0
Label : 79, Area : 1.0
Label : 80, Area : 1.0
Label : 81, Area : 1.0
Label : 82, Area : 1.0
Label : 83, Area : 1.0
```

In [ ]:
```
props = measure.regionprops_table(label_image, image,
                                   properties=['label', 'area', 'equivalent_diameter',
                                               'mean_intensity', 'solidity'])
```

In [ ]:
```
df = pd.DataFrame(props)
df.head()
```

Out[ ]:

	label	area	equivalent_diameter	mean_intensity	solidity
0	1	1.0	1.128379	121.0	1.0
1	2	1.0	1.128379	118.0	1.0
2	3	1.0	1.128379	120.0	1.0
3	4	2.0	1.595769	128.5	1.0
4	5	2.0	1.595769	121.5	1.0

In [ ]:
```
df = df[df['area'] > 50]
print(df.head())
```

```
    label   area  equivalent_diameter  mean_intensity  solidity
18     19   55.0             8.368284      166.909091  0.544554
46     47   77.0             9.901487      158.103896  0.700000
51     52   74.0             9.706685      187.148649  0.596774
57     58   94.0            10.940042      173.648936  0.630872
63     64  293.0            19.314740      168.259386  0.664399
```

In [ ]:
```
scale = 2

df['area_no_units'] = df['area'] * (scale)
df['equivalent_diameter_microns'] = df['equivalent_diameter'] * (scale)

print(df.head())
```

```
df.to_csv('cleaned_data_1.csv')
```

```
    label   area  equivalent_diameter  mean_intensity  solidity  \
18     19   55.0             8.368284      166.909091  0.544554
46     47   77.0             9.901487      158.103896  0.700000
51     52   74.0             9.706685      187.148649  0.596774
57     58   94.0            10.940042      173.648936  0.630872
63     64  293.0            19.314740      168.259386  0.664399

    area_no_units  equivalent_diameter_microns
18          110.0                    16.736568
46          154.0                    19.802974
51          148.0                    19.413369
57          188.0                    21.880084
63          586.0                    38.629480
```