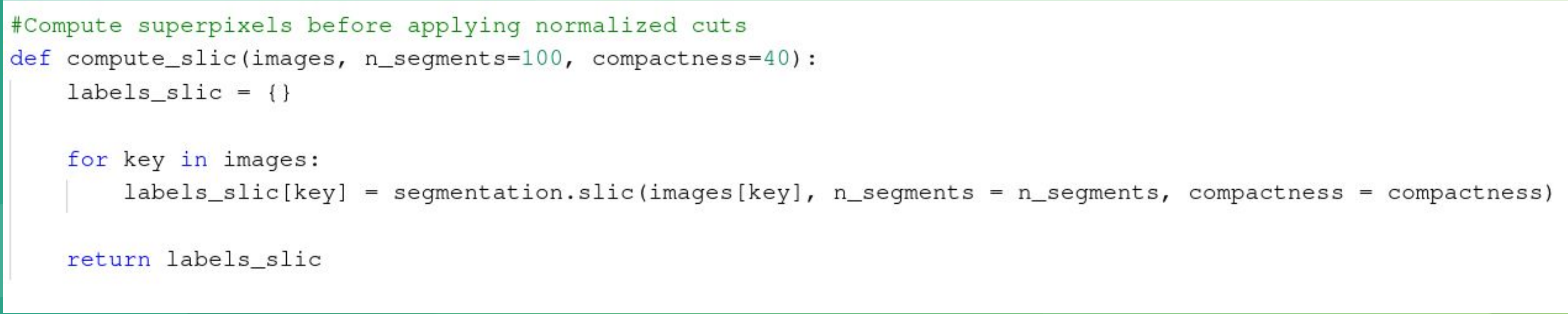
Image segmentation

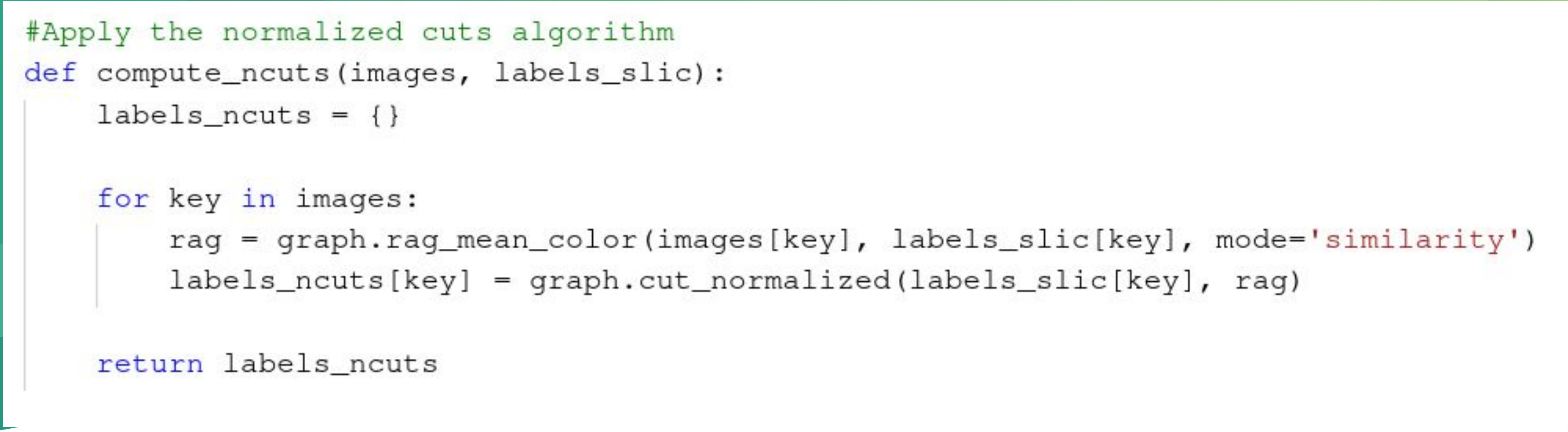
Assignment 4

Luigi
Quarantiello

Normalized cuts - Code snippets

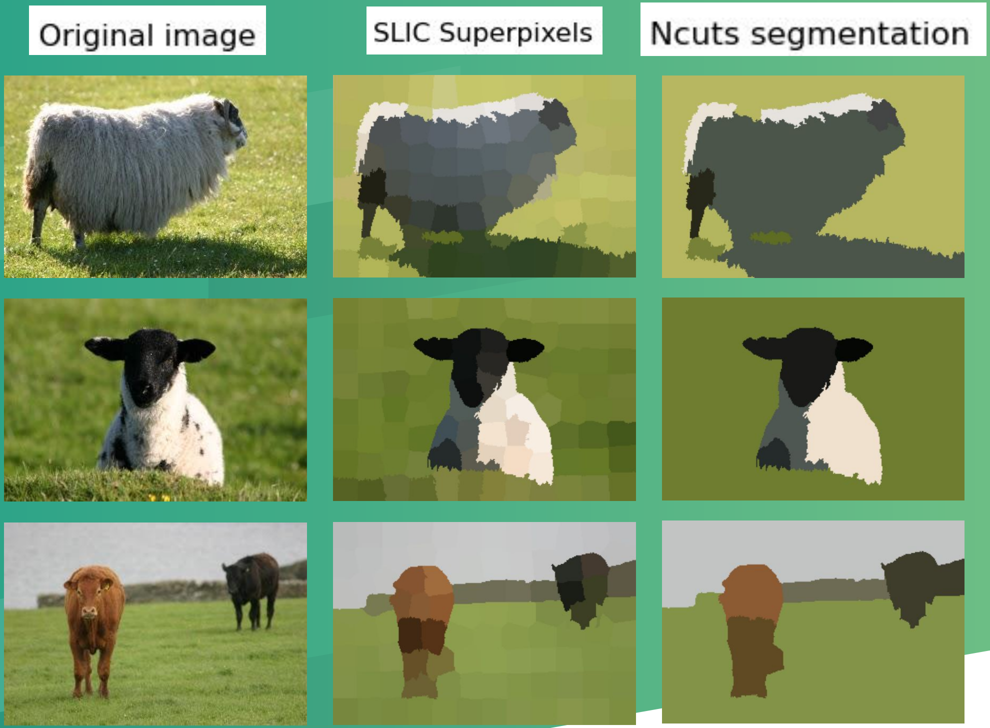


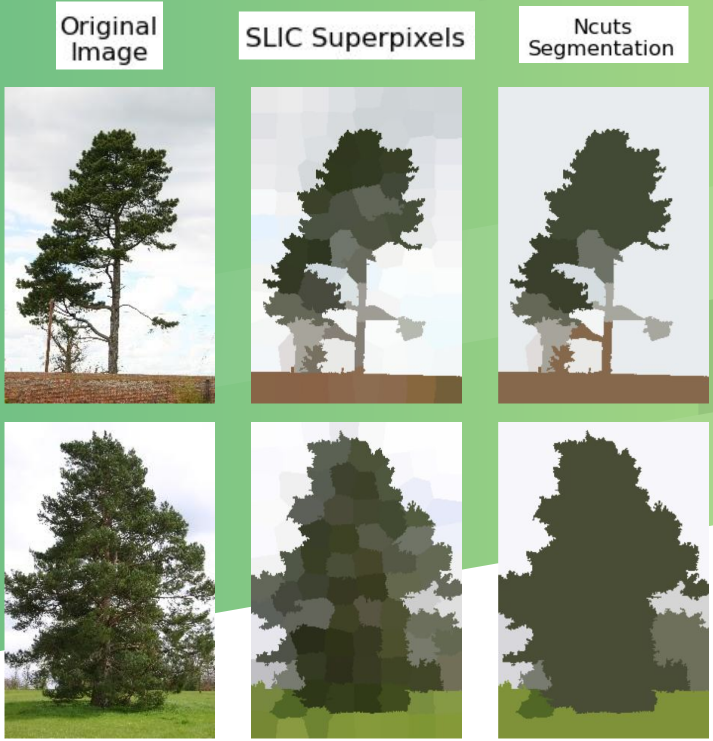
```python
#Compute superpixels before applying normalized cuts
def compute_slic(images, n_segments=100, compactness=40):
    labels_slic = {}

    for key in images:
        labels_slic[key] = segmentation.slic(images[key], n_segments = n_segments, compactness = compactness)

    return labels_slic
```



```python
#Apply the normalized cuts algorithm
def compute_ncuts(images, labels_slic):
    labels_ncuts = {}

    for key in images:
        rag = graph.rag_mean_color(images[key], labels_slic[key], mode='similarity')
        labels_ncuts[key] = graph.cut_normalized(labels_slic[key], rag)

    return labels_ncuts
```

Normalized cuts - Results

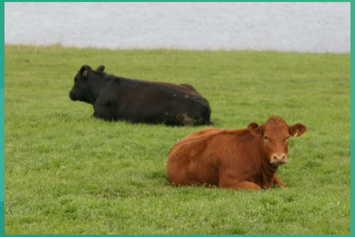




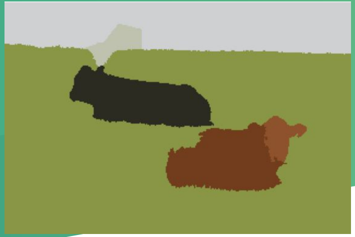
K-means segmentation - Code and comparison

```python
#Compute image segmentation using k-means algorithm
def compute_kmeans(images, k=3, criteria=(cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 10, 1.0),
                   attempts=10, flag=cv2.KMEANS_RANDOM_CENTERS):
    centers = {}
    labels = {}
    for key in images:
        img = images[key].reshape((-1,3))
        img = np.float32(img)

        _,labels[key],centers[key] = cv2.kmeans(img, k, None, criteria, attempts, flag)

    return labels, centers
```

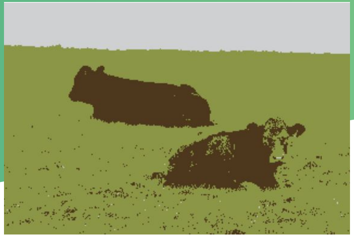
Original image



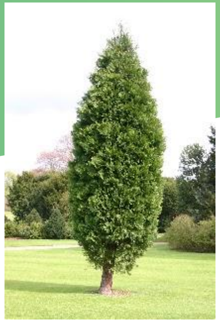
Ncuts segmentation



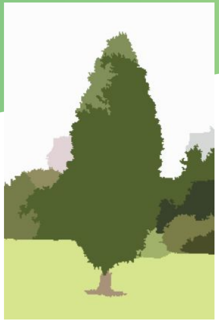
K-means Segmentation



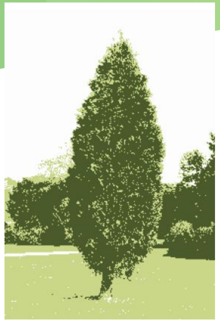
Original Image



Ncuts Segmentation



K-means Segmentation



Final considerations

Ncuts segmentation

- Produces results with a clear distinction between the various objects
- Can be computationally expensive, but has good performances even with a small number of superpixels

K-means segmentation

- Faster than the ncuts algorithm, but the results are less clear