CS421

Lab5

Image Segmentation using K-Means and normalized cut algorithm

Team Members:

- Ahmed Khalil Yakout (7)
- Omar Shawky (43)

Introduction

We intend to perform image segmentation. Image segmentation means that we can group similar pixels together and give these grouped pixels the same label. The grouping problem is a clustering problem. We want to study the use of K-means and Normalized-Cut methods on the Berkeley Segmentation Benchmark. Below we will show the needed steps to achieve the goal of the assignment, our results, evaluation and comparison between different approaches.

We used python in the implementation of this lab.

Dataset

- Berkeley Segmentation Benchmark
- The datasets contains 500 images, the test set is 200 images only and we will report our results on test set only.

The function `"visualize_image"` reads an image and display an image with its associated ground truth segmentation(s).

- Other sample runs can be found in the jupyter file.

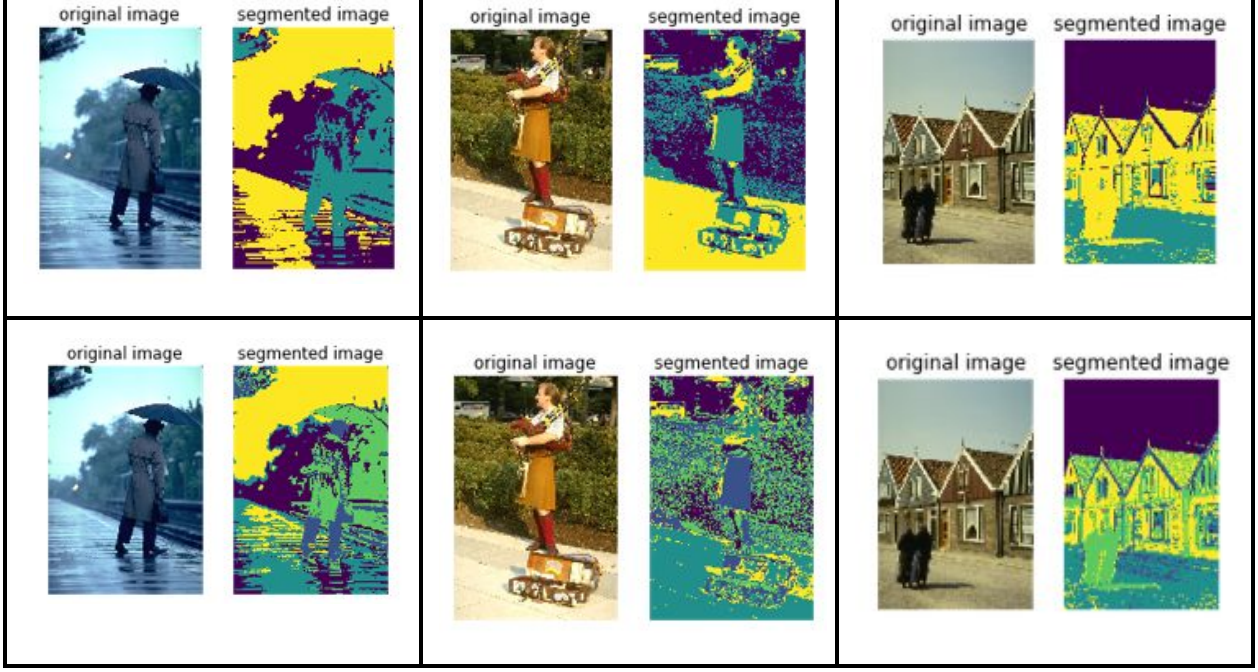
K-Means (our implementation)

Implementation

In K-Means we first chose random K centroids from our data points, then calculate the distance between every data point and the centroids, after that we find the closest the centroid and assign the data point to it.

To update the centroid we find the mean for the new cluster points and do the same previous steps until we no longer can update centroids or reached the max number of iteration.

Sample Runs (With different values of K)




original image
segmented image
original image
segmented image
original image
segmented image
original image
segmented image
original image
segmented image
original image
segmented image
original image
segmented image
original image
segmented image
original image
segmented image

Evaluation

We use F-Measure and Conditional Entropy to evaluate the results of clustering.

- F1-Score
    - average value of f1 at K = 3 is: avg: 1.3727344, max: 1.7611743, min: 0.7229798
      average value of f1 at K = 5 is: avg: 1.0763230, max: 1.4730343, min: 0.6069716
      average value of f1 at K = 7 is: avg: 0.8411452, max: 1.1545390, min: 0.4280017
      average value of f1 at K = 9 is: avg: 0.7129133, max: 0.9625061, min: 0.3557265
      average value of f1 at K = 11 is: avg: 0.583774, max: 0.7737563, min: 0.2888760
- Conditional Entropy
    - average value of cond_entropy at K = 3 is: avg: 0.0259, max: 0.0333, min: 0.0164
      average value of cond_entropy at K = 5 is: avg: 0.0421, max: 0.0556, min: 0.0252
      average value of cond_entropy at K = 7 is: avg: 0.0578, max: 0.0759, min: 0.0337
      average value of cond_entropy at K = 9 is: avg: 0.0726, max: 0.0941, min: 0.0425
      average value of cond_entropy at K = 11 is: avg: 0.084, max: 0.1117, min: 0.0475

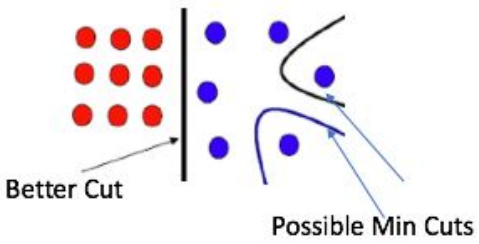
Normalized Cut

Little Background

The idea of normalized cut is based on the famous problem min cut, which given a graph and weight on edges we partition the graph into k sets with minimum cost. The cost here is the sum of weights on the edges connecting the sets together so assume we partition into only two sets A and B our cost is:

$$\text{cut}(A, B) := \sum_{i \in A, j \in B} w_{ij}$$

and we saw earlier in the Algorithm course how we can be solve it easily with max flow algorithm in O(VE) time, the only problem with min cut, it does not split the graph with equal nodes in each subgraph.



Normalized cut algorithm try to solve this problem so that sizes of set A and B are similar and the cost will be:

$$\text{Ncut}(A, B) := \text{cut}(A, B)\left(\frac{1}{\text{vol}(A)} + \frac{1}{\text{vol}(B)}\right)$$

$$\text{vol}(A) = \sum_{i \in A} d_i$$

Algorithm

ALGORITHM 16.1. Spectral Clustering Algorithm

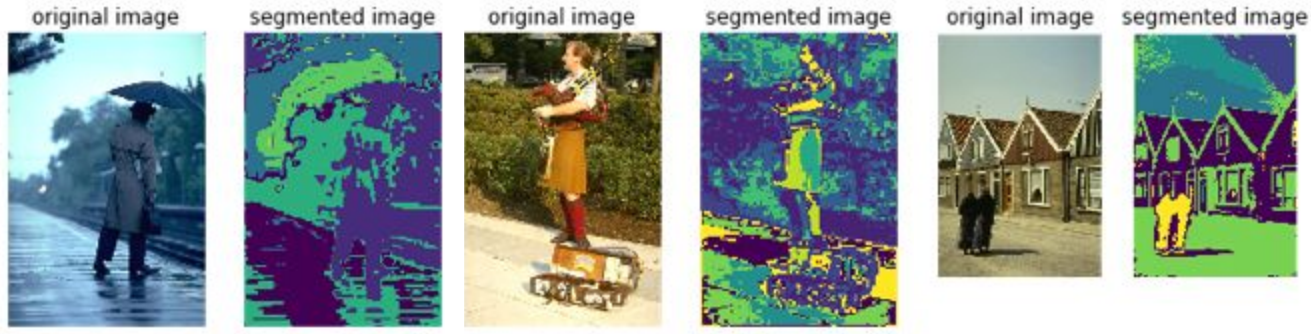
SPECTRAL CLUSTERING $(\mathbf{D}, k)$:

1 Compute the similarity matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$
2 **if** *ratio cut* **then** $\mathbf{B} \leftarrow \mathbf{L}$
3 **else if** *normalized cut* **then** $\mathbf{B} \leftarrow \mathbf{L}^s$ or $\mathbf{L}^a$
4 Solve $\mathbf{B}\mathbf{u}_i = \lambda_i \mathbf{u}_i$ for $i = n, \ldots, n-k+1$, where $\lambda_n \leq \lambda_{n-1} \leq \cdots \leq \lambda_{n-k+1}$
5 $\mathbf{U} \leftarrow \begin{pmatrix} \mathbf{u}_n & \mathbf{u}_{n-1} & \cdots & \mathbf{u}_{n-k+1} \end{pmatrix}$
6 $\mathbf{Y} \leftarrow$ normalize rows of $\mathbf{U}$ using Eq. (16.19)
7 $\mathcal{C} \leftarrow \{C_1, \ldots, C_k\}$ via K-means on $\mathbf{Y}$

- For simplicity and time constraints we use the implementation of "*scikit-learn*" library for Spectral Clustering algorithm.
    - https://scikit-learn.org/stable/modules/generated/sklearn.cluster.SpectralClustering.html
- Some issues:
    - Choice of K (We used different value of Ks: 3, 5, 7, 9, 11)
    - Choice of similarity matrix (We used 5-NN graph)
    - Choice of kernel and gamma value for the choice of gaussian kernel (We used RBF kernel with different values of gammas = {1, 10})
    - Complexity of the algorithm O(n^2 * d) to find the similarity matrix and O(n^3) for eigenvalue decomposition.

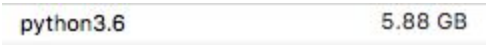
Sample Runs (With different values of K)

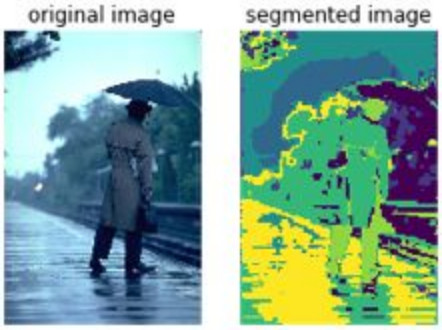
Using 5-NN graph



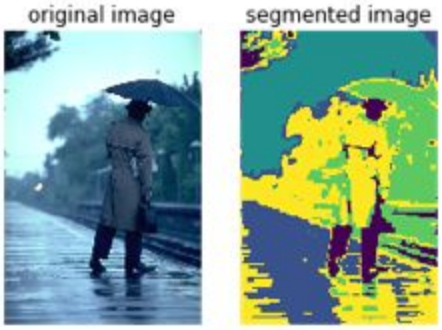
Using RBF kernel and gamma = 1

It took very long time so we only trained on one image :)

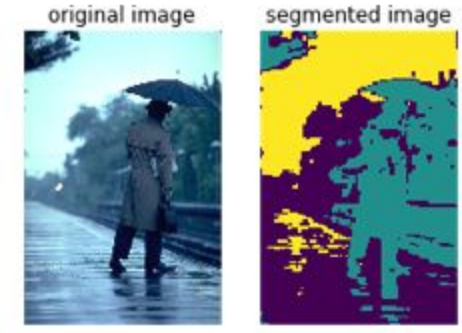




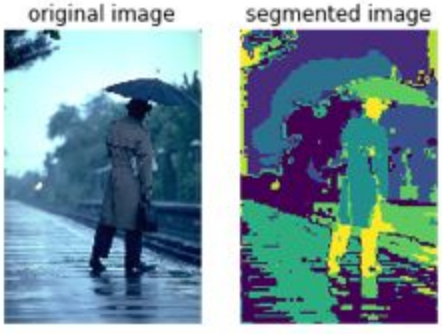
K = 3



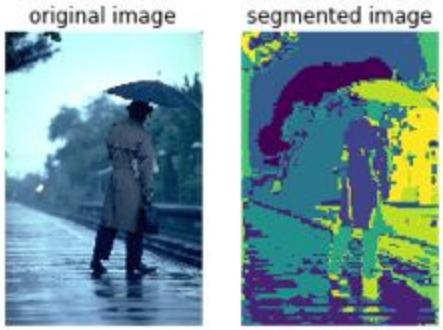
K = 5



K = 7



K = 9



K = 11

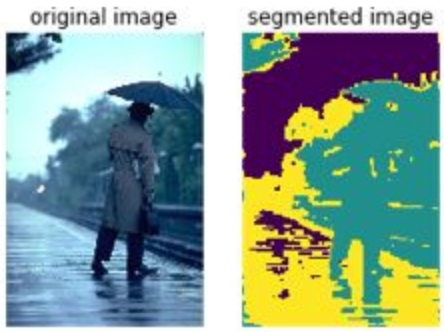
F1-Scores

K = 3: avg: 0.519, max: 0.515, min: 0.515
K = 5: avg: 0.297, max: 0.298, min: 0.298
K = 7: avg: 0.197, max: 0.197, min: 0.197
K = 9: avg: 0.166, max: 0.162, min: 0.162
K = 11: avg: 0.132, max: 0.132, min: 0.132

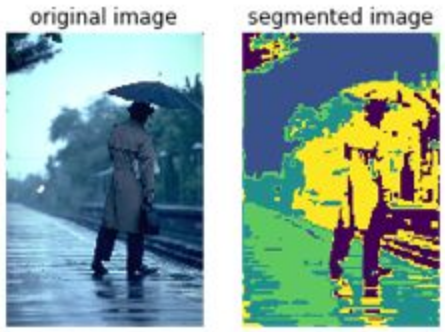
Conditional Entropy

K = 3: avg: 0.0431, max: 0.0431, min: 0.043
K = 5: avg: 0.0652, max: 0.0652, min: 0.065
K = 7: avg: 0.0917, max: 0.0917, min: 0.091
K = 9: avg: 0.1126, max: 0.1126, min: 0.112
K = 11: avg: 0.1379, max: 0.137, min: 0.137

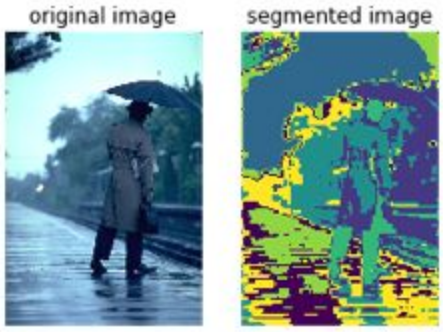
Using RBF kernel and gamma = 10



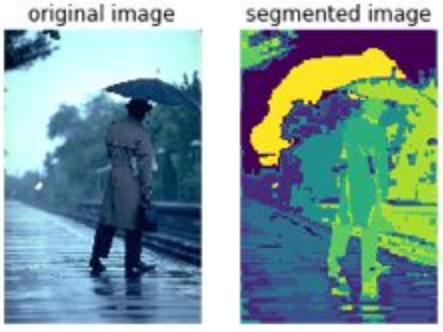
K = 3



K = 5



K = 7



K = 9

K = 11

F1-Score
K = 3: avg: 0.303, max: 0.516, min: 0.084
K = 5: avg: 0.185, max: 0.301, min: 0.050
K = 7: avg: 0.138, max: 0.229, min: 0.036
K = 9: avg: 0.109, max: 0.186, min: 0.028
K = 11: avg: 0.089, max: 0.155, min: 0.023

Conditional Entropy
K = 3 is: avg: 0.0228, max: 0.0511, min: 0.0
K = 5 is: avg: 0.0363, max: 0.0859, min: 0.0
K = 7 is: avg: 0.0479, max: 0.1127, min: 0.0
K = 9 is: avg: 0.0598, max: 0.1390, min: 0.0
K = 11 is: avg: 0.0722, max: 0.161, min: 0.0

k-means vs. Spectral clustering

- Applying k-means to Laplacian eigenvectors allows us to find cluster with non-convex boundaries.

- For smaller values of k the k-means algorithm give good results. For larger values of k, the segmentation is very coarse, many clusters appear in the images at discrete places. This is because Euclidean distance is not a very good metric for segmentation processes.

- Better algorithms like the graph based Normalized cuts give good results for larger value of k. One basic difference between normalized cuts and k-means algorithm is that k-means algorithm consider pixels with relatively close intensity values as belonging to one segment, even if it is not locationally close. Normalized cuts considers such areas as separate segments. Normalize cuts implementation is computationally complex. The eigenvalue method take a long time for a full scale image. So images have to be resized to get faster results as we did in the code.

Evaluation

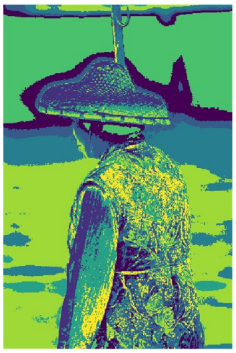
- Conditional Entropy:
    - average value of cond_entropy at K = 3 is: avg: 0.021408, max: 0.048789, min: 0.0
      average value of cond_entropy at K = 5 is: avg: 0.030691, max: 0.065796, min: 0.0
      average value of cond_entropy at K = 7 is: avg: 0.032369, max: 0.082325, min: 0.0
      average value of cond_entropy at K = 9 is: avg: 0.050156, max: 0.105026, min: 0.0
      average value of cond_entropy at K = 11 is: avg: 0.06000, max: 0.134924, min: 0.0

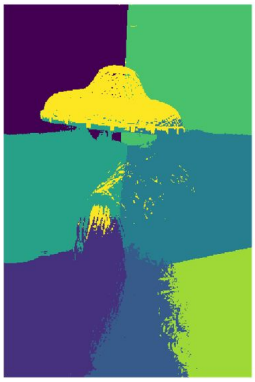
- F1-Score
    - average value of f1 at K = 3 is: avg: 0.2323339, max: 0.3908406, min: 0.084463
      average value of f1 at K = 5 is: avg: 0.1917094, max: 0.3012482, min: 0.050678
      average value of f1 at K = 7 is: avg: 0.1083356, max: 0.2137811, min: 0.036198
      average value of f1 at K = 9 is: avg: 0.1038105, max: 0.1655257, min: 0.028154
      average value of f1 at K = 11 is: avg: 0.088127, max: 0.1521786, min: 0.023035

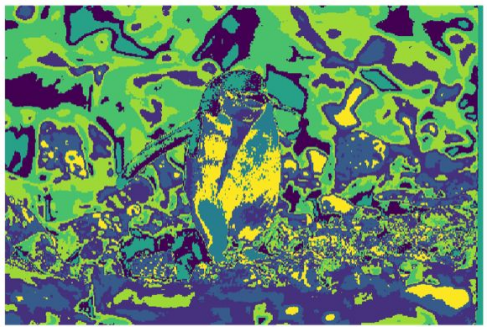
Bonus

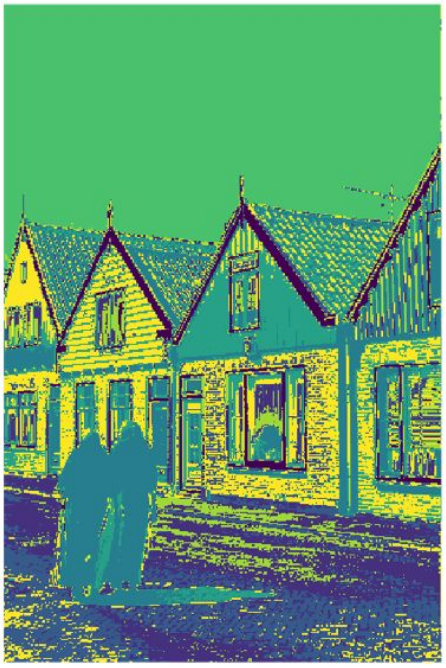
K-means without spatial

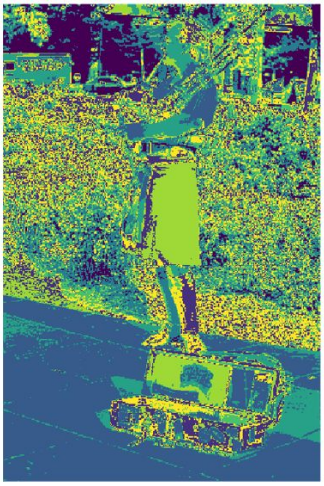
K-means with spatial

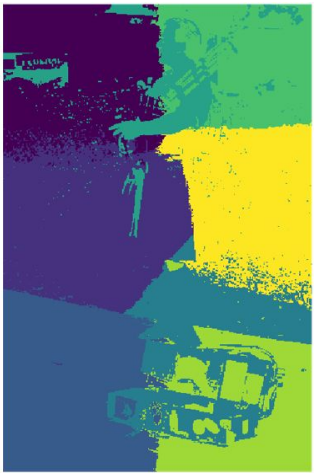












```
def kmeans_segment(img, n_clusters=DEFAULT_N_CLUSTERS,
                        max_iter=K_MEANS_DEFAULT_MAX_ITER,
                        include_spatial=False,
                        visualize=False):
    n = img.shape[0]
    m = img.shape[1]

    if include_spatial:
        xx = np.arange(n)
        yy = np.arange(m)
        X, Y = np.meshgrid(yy, xx)
        img = np.concatenate((Y.reshape(n, m, 1), X.reshape(n, m, 1), img), axis=2)
        print("kmeans_segment(:include_spatial) img.shape = {}".format(img.shape))

    # we do img.shape[-1] so we get last shape dim which in case of
    # include_spatial=True it will be 5 and in case of include_spatial=False
    # it will be # colors which is RGB = 3
    img = img.reshape(-1, img.shape[-1]) # 2D array (n*m, features_count)

    segmented_image = KMeans(n_clusters, max_iter).fit(img).reshape(n, m)

    if visualize:
        plt.figure(figsize=(12, 12))
        plt.axis('off')
        plt.imshow(segmented_image)

    return segmented_image
```

XX, YY = meshgrid(x, y)

```
   7                1 7   2 7   3 7   4 7        1 2 3 4    7 7 7 7
y  6          >>    1 6   2 6   3 6   4 6   >>   1 2 3 4    6 6 6 6
   5                1 5   2 5   3 5   4 5        1 2 3 4    5 5 5 5
     1 2 3 4                                       XX         YY
       x
```

Big Picture

- Applying k-means segmentation on set of 5 images with K = 5.

  Image (1)

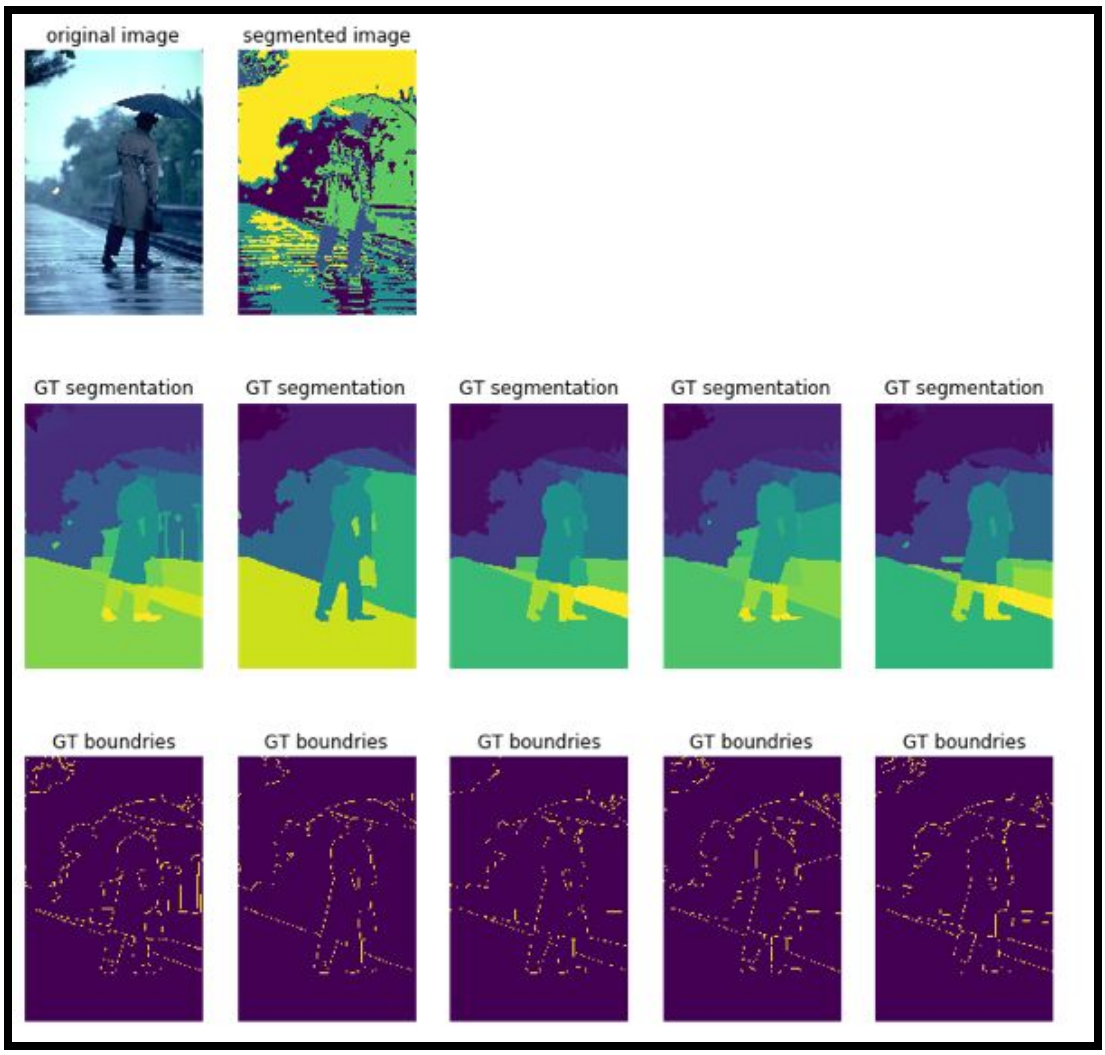


Image (2)

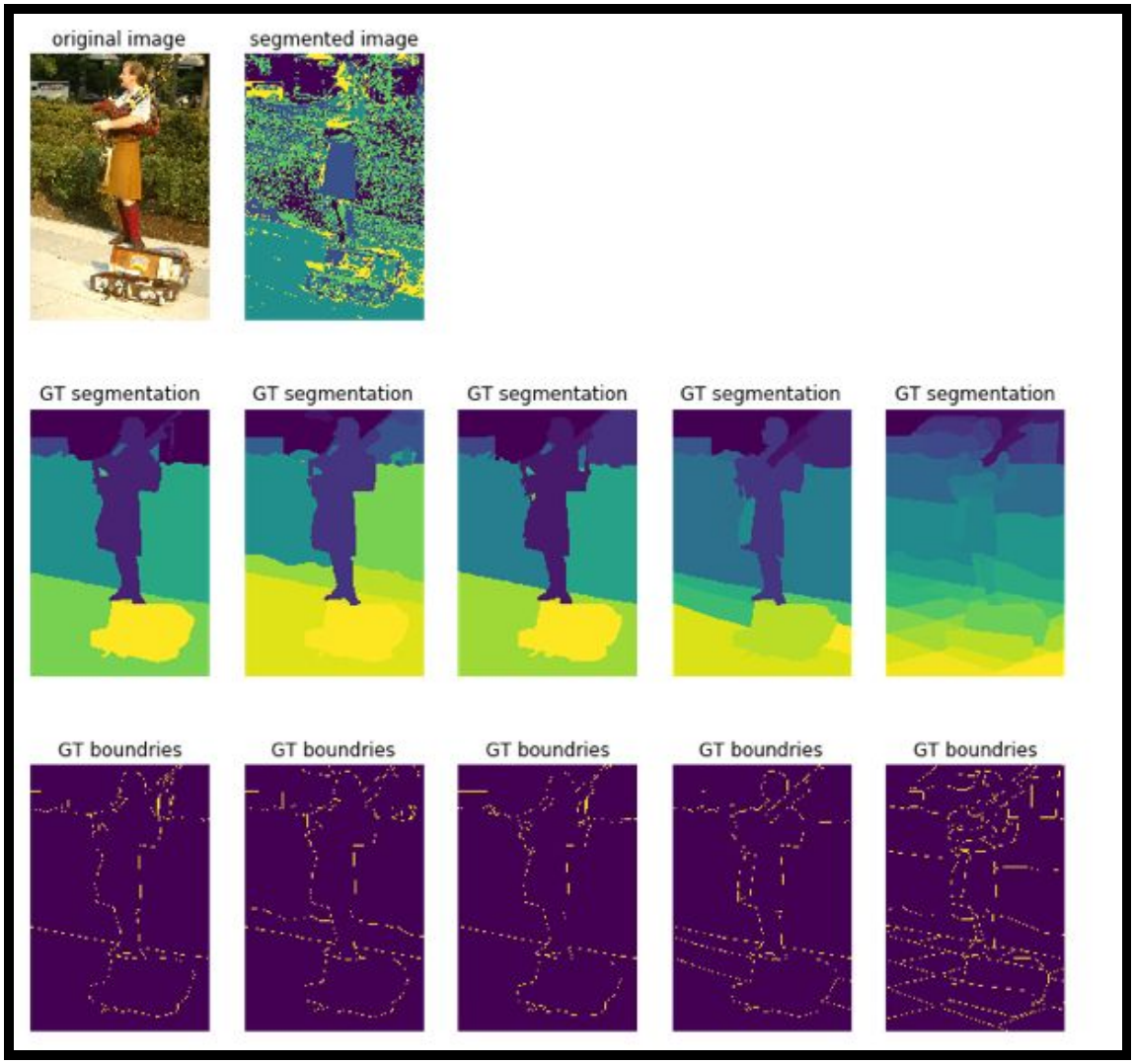

original image
segmented image
GT segmentation
GT segmentation
GT segmentation
GT segmentation
GT segmentation
GT boundries
GT boundries
GT boundries
GT boundries
GT boundries

Image (3)

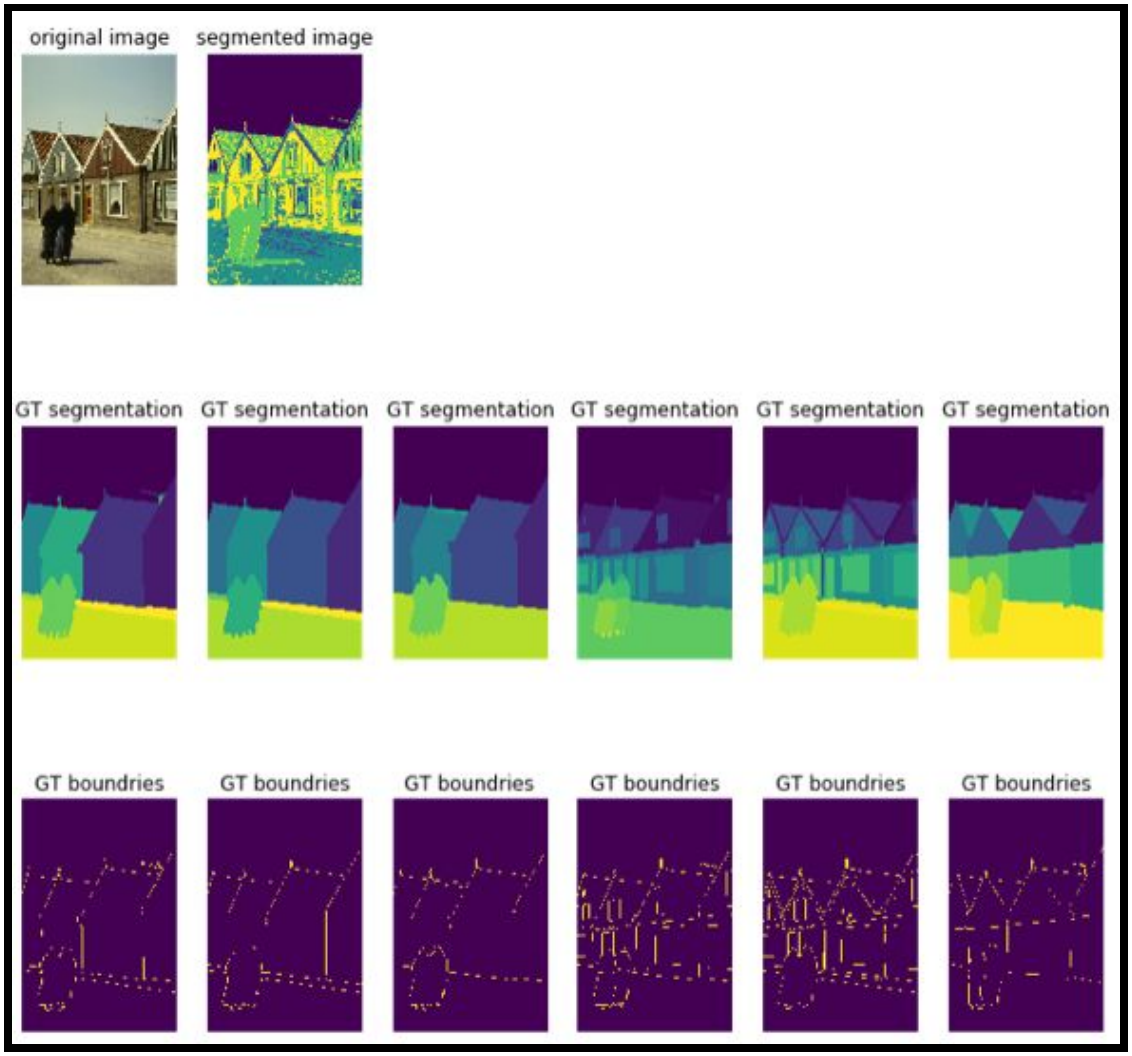

original image
segmented image
GT segmentation
GT segmentation
GT segmentation
GT segmentation
GT segmentation
GT segmentation
GT boundries
GT boundries
GT boundries
GT boundries
GT boundries
GT boundries

Image (4)

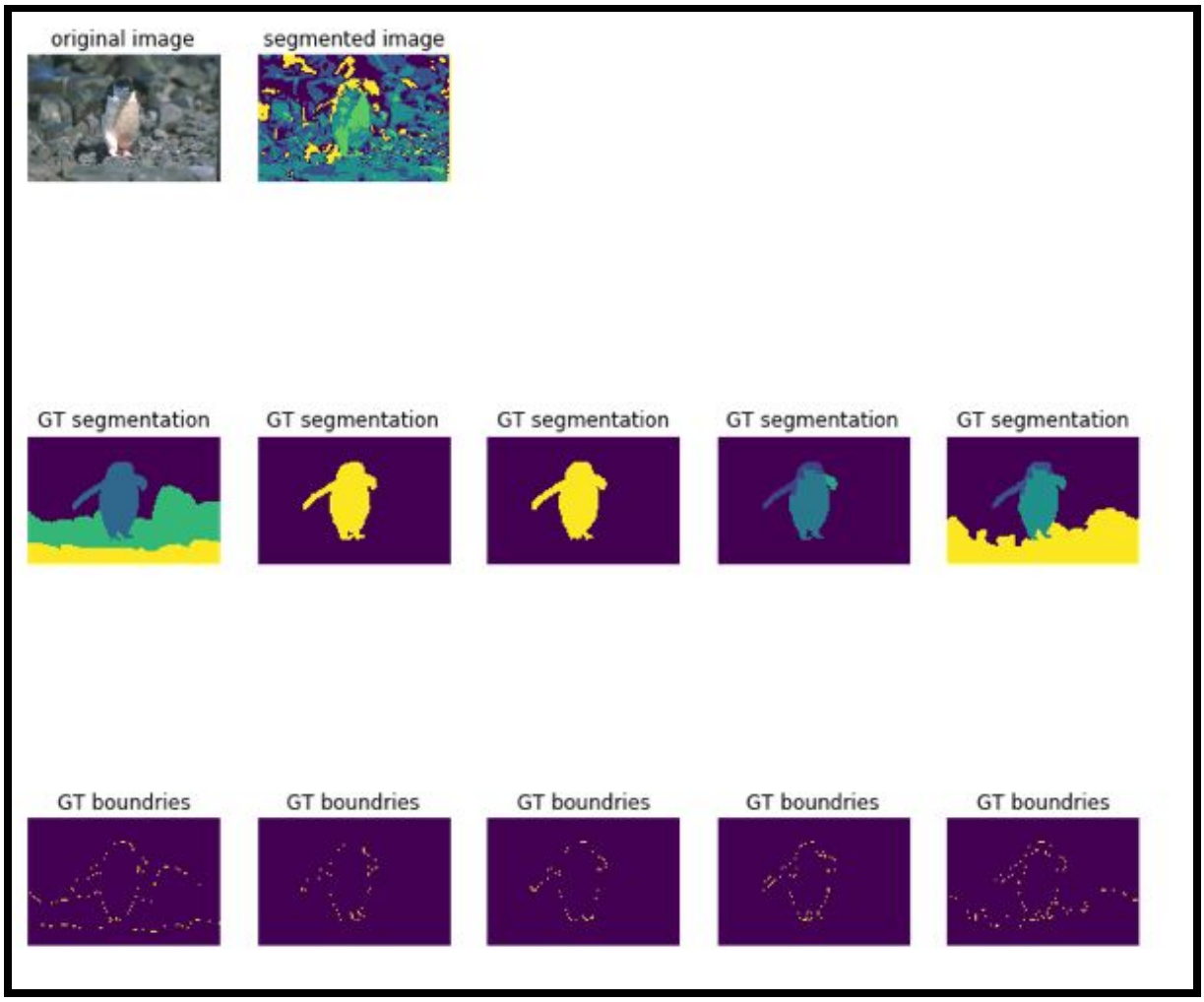
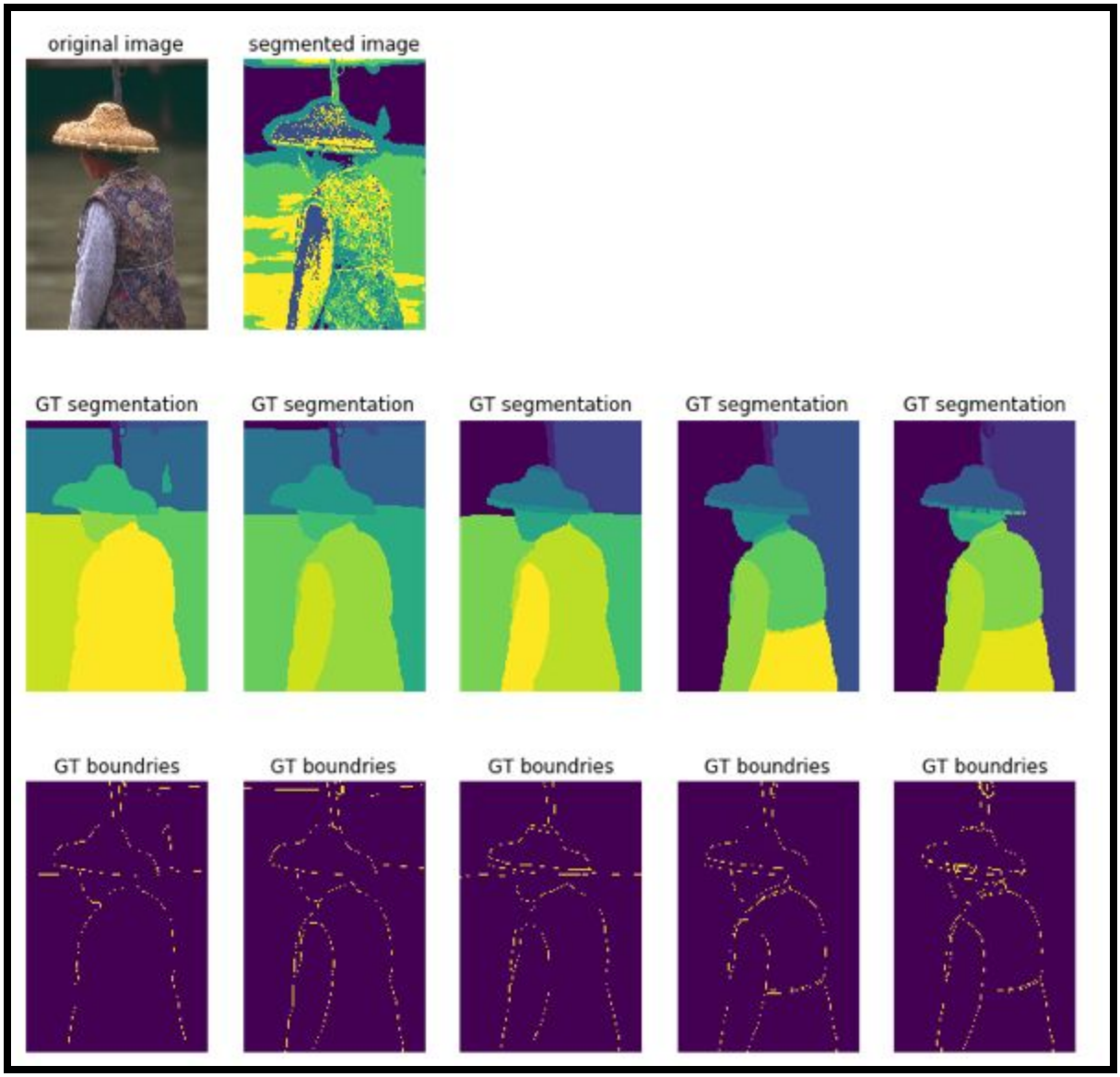

original image
segmented image
GT segmentation
GT segmentation
GT segmentation
GT segmentation
GT segmentation
GT boundries
GT boundries
GT boundries
GT boundries
GT boundries

Image (5)


original image
segmented image
GT segmentation
GT segmentation
GT segmentation
GT segmentation
GT segmentation
GT boundries
GT boundries
GT boundries
GT boundries
GT boundries

- Applying Normalized cut segmentation on set of 5 images with K = 5 and 5-NN graph.

Image (1)

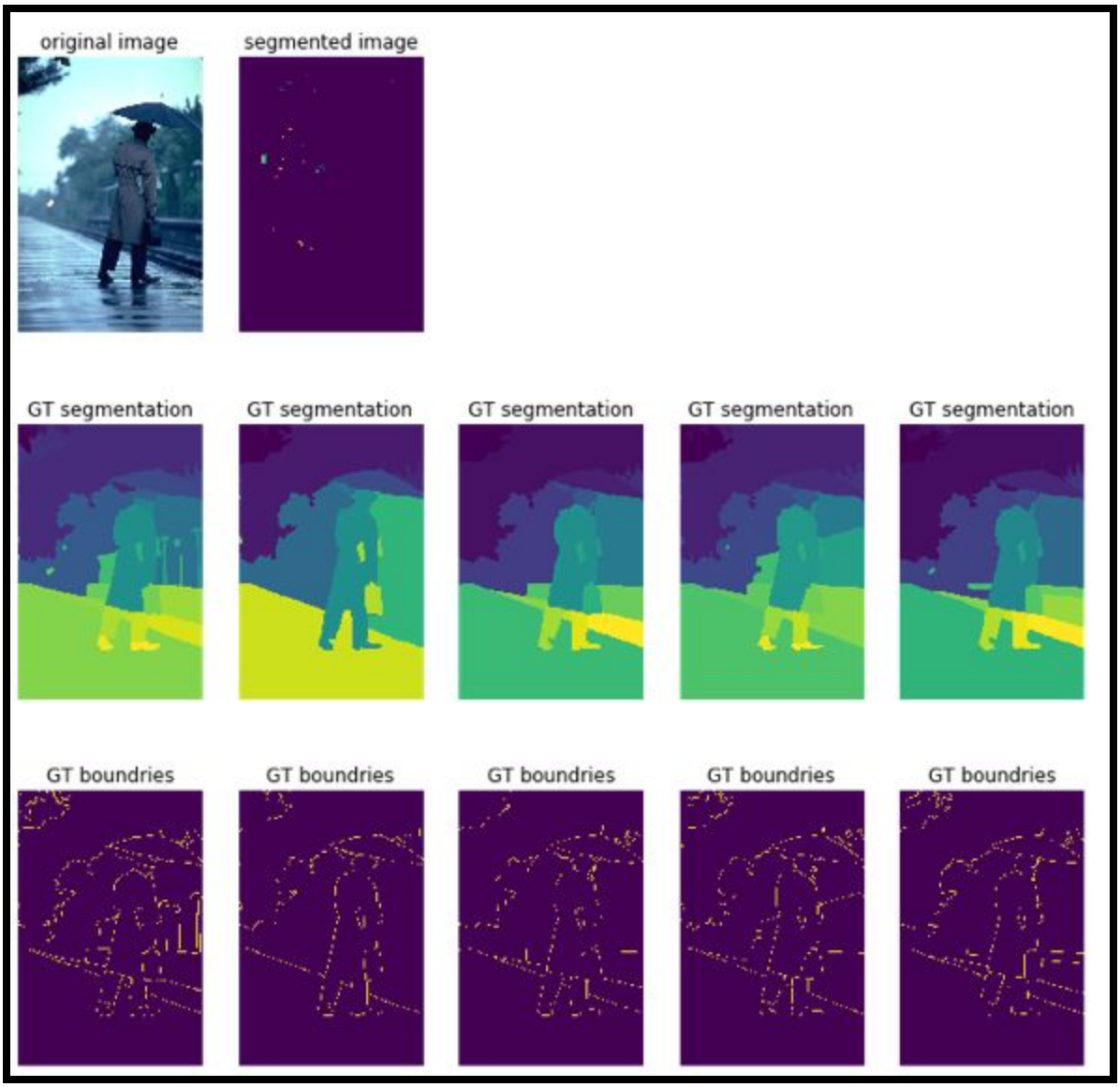


Image (2)

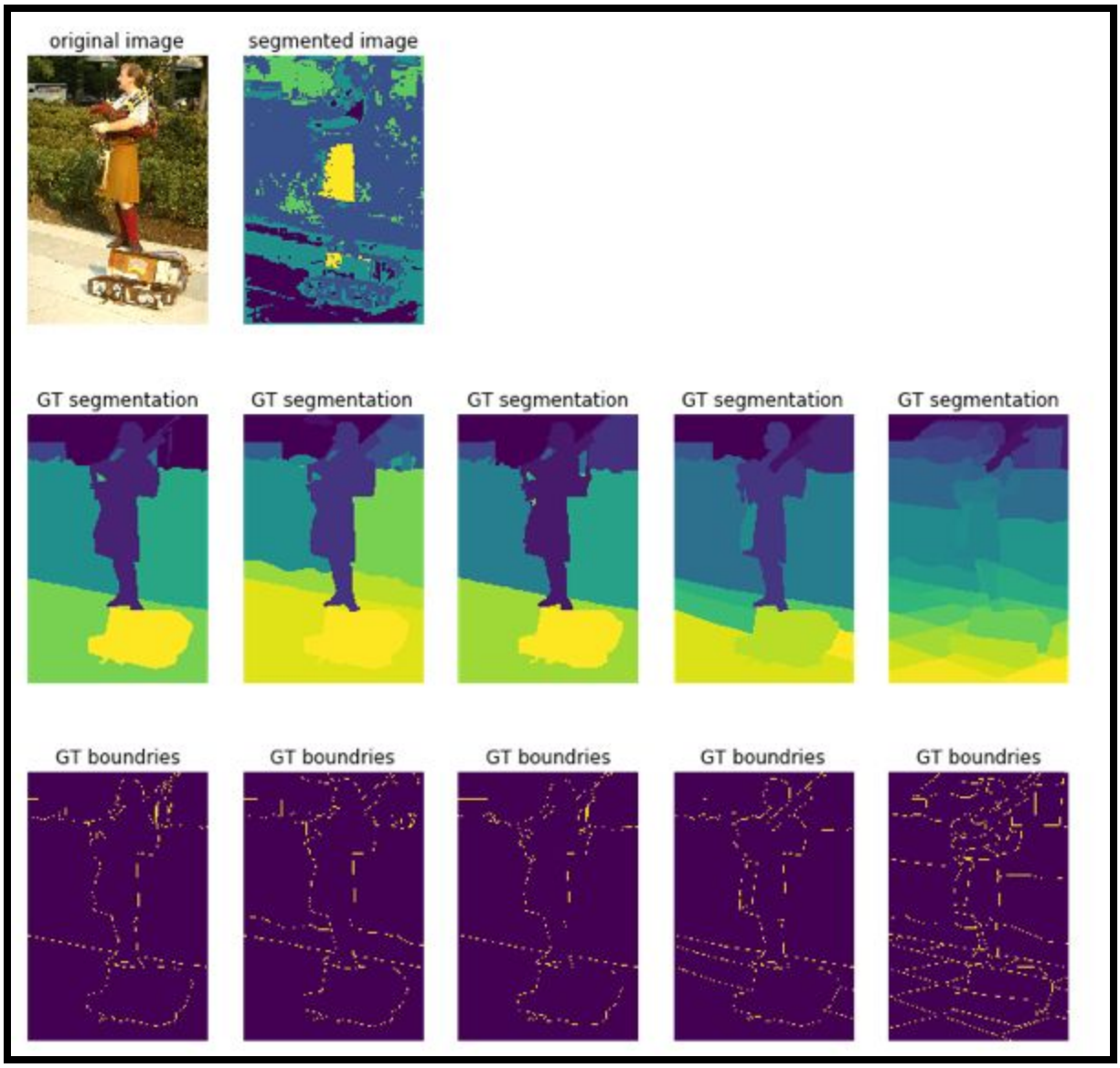

original image
segmented image
GT segmentation
GT segmentation
GT segmentation
GT segmentation
GT segmentation
GT boundries
GT boundries
GT boundries
GT boundries
GT boundries

Image (3)

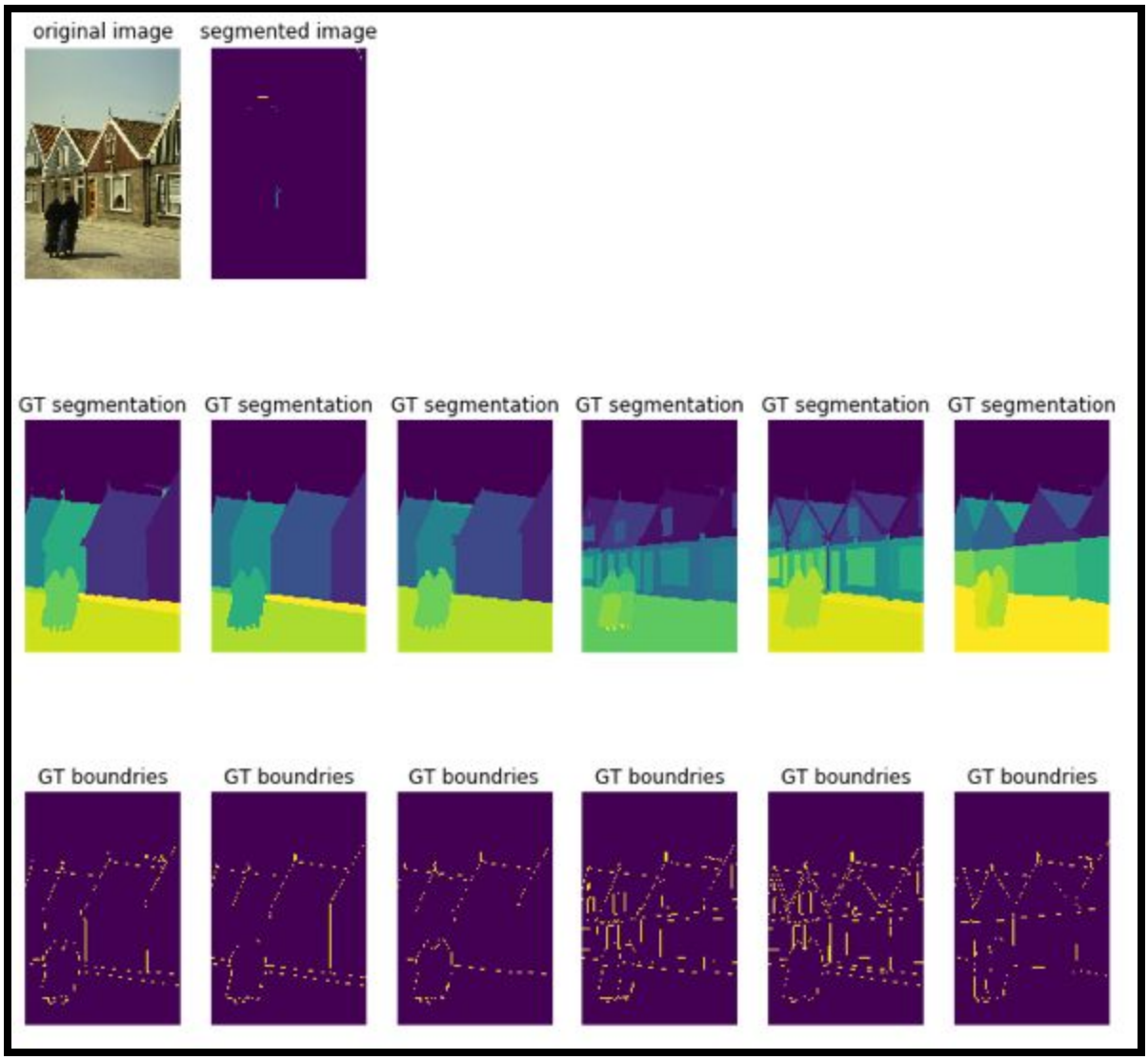

original image
segmented image
GT segmentation
GT segmentation
GT segmentation
GT segmentation
GT segmentation
GT segmentation
GT boundries
GT boundries
GT boundries
GT boundries
GT boundries
GT boundries

Image (4)

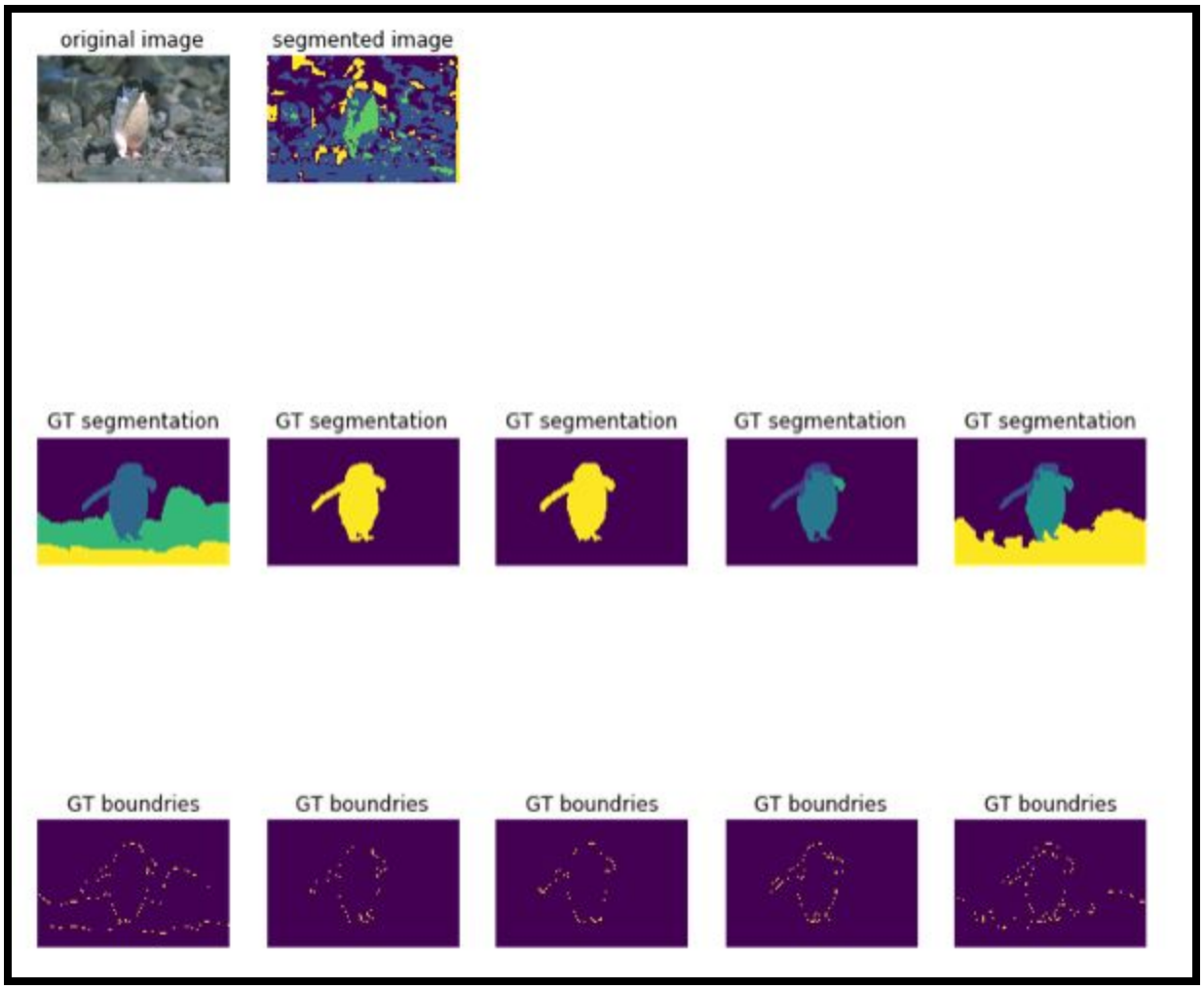
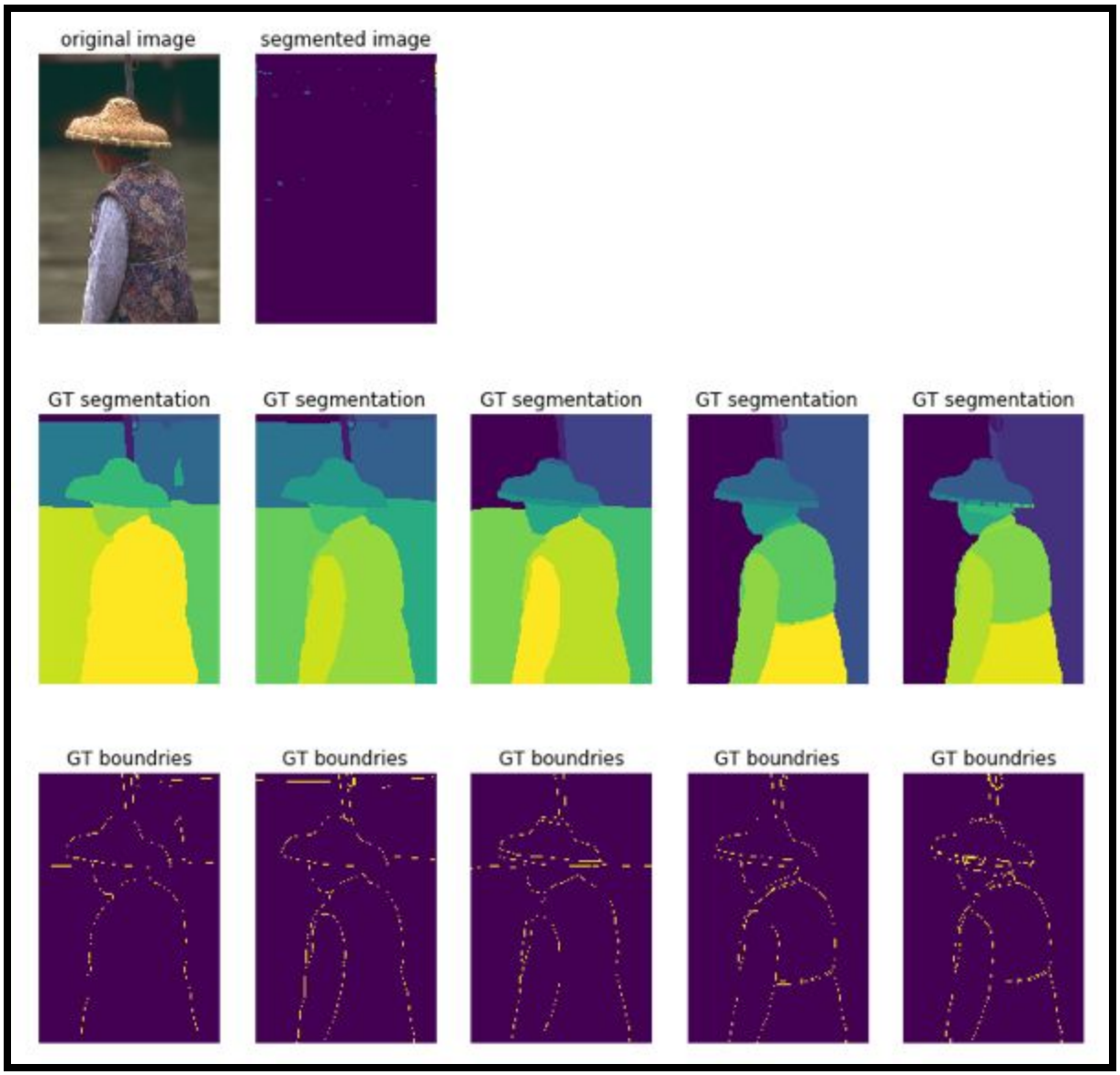

original image
segmented image
GT segmentation
GT segmentation
GT segmentation
GT segmentation
GT segmentation
GT boundries
GT boundries
GT boundries
GT boundries
GT boundries

Image (5)


original image
segmented image
GT segmentation
GT segmentation
GT segmentation
GT segmentation
GT segmentation
GT boundries
GT boundries
GT boundries
GT boundries
GT boundries