HW 4

Ajit Sarkaar

PART 1

SLIC

(a) Explain your distance function for measuring the similarity between a pixel and cluster in the 5D space.

The distance function used for measuring similarity D is given below:

$$D = \sqrt{{d_c}^2 + \left(\frac{d_s}{S}\right)^2 m^2}.$$

Where m = compactness factor, which determines the influence of spatial distances in the overall distance measure,

S = number of cluster centers,

$d_c = \sqrt{(l_j - l_i)^2 + (a_j - a_i)^2 + (b_j - b_i)^2}$, D_c is the sum of squared differences in the LAB color space and,

$d_s = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2}$, D_s is the sum of squared differences of the spatial coordinates.

(b) Choose one image, try three different weights on the color and spatial feature and show the three segmentation results. Describe what you observe.

Original Image

Compactness = 5

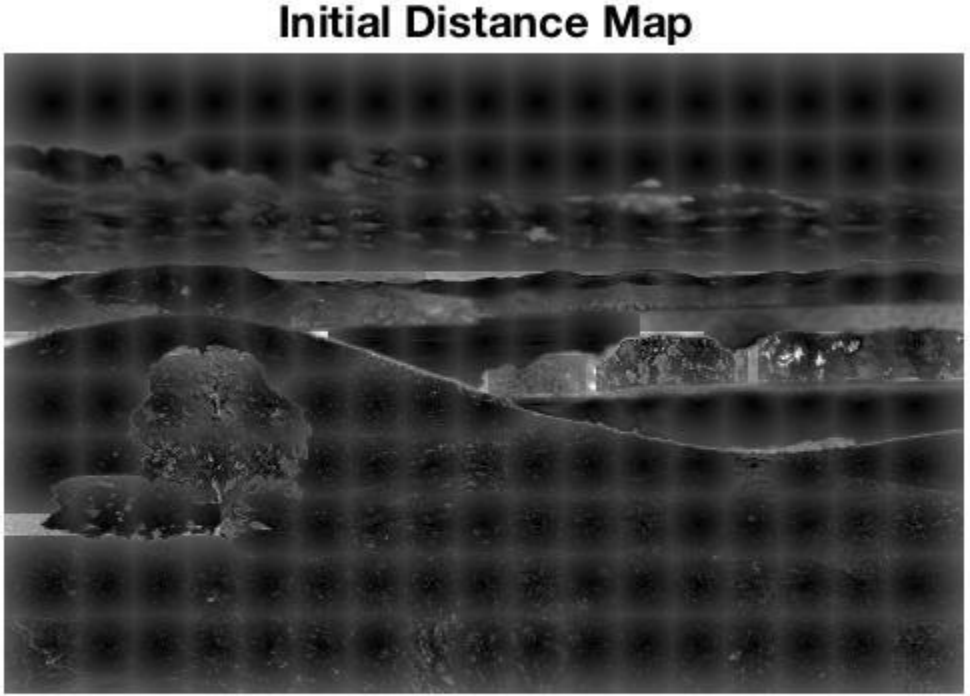
Compactness = 25

Compactness = 40

Higher compactness means more weight is given to spatial features than color features. Therefore, as compactness is increases, minor variations in color over a small area does not create a separate cluster, but it is included in the same cluster which is closer spatially. Therefore, only large changes in color form a cluster boundary, instead of many small clusters. As seen in the examples, image with compactness = 5 has many small clusters in the lower half, whereas, these small clusters go away in the image where compactness = 40, as more weight is given to spatial distances and only large changes in color produce a boundary.

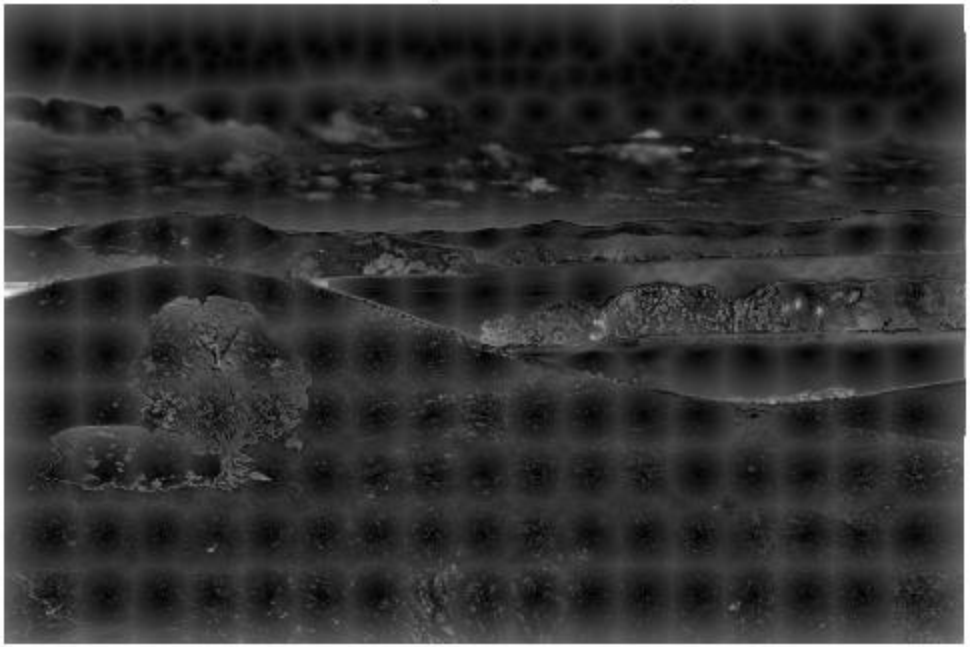
(c) Choose one image, show the error map (1) at the initialization and (2) at convergence. Note: error - distance to cluster center in the 5D space.

Original Image



Error map at initialization in 5 D space(L, A, B, Height, Width)

Distance Map At Convergence



Error map at convergence in 5 D space(L, A, B, Height, Width)

(d) Choose one image, show three superpixel results with different number of K, e.g., 64, 256, 1024 and run time for each K (use tic, toc)

Original Image

K = 64

Runtime: 4.3145 seconds.

K = 256

Runtime: 6.2458 seconds.

K = 1024

Runtime: 8.6633 seconds.

(e) Run your algorithms on the subset (50 images) of Berkeley Segmentation Dataset (BSD) and report two types metrics (1) boundary recall and (2) under-segmentation error with K = 64, 256 and 1024. Also, report averaged run time per image for the BSD

K = 64

```
executing:    [S, time, imgVis] = slic(I, 64, 25)
working on: 1 (name: 10081)
   3.0649
working on: 2 (name: 14085)
   3.0804
working on: 3 (name: 14092)
   2.9081
working on: 4 (name: 15011)
   2.9600
working on: 5 (name: 15062)
   2.9485
working on: 6 (name: 16004)
   3.0284
working on: 7 (name: 16068)
   2.9629
working on: 8 (name: 17067)
   3.0109
working on: 9 (name: 20069)
   3.1118
working on: 10 (name: 2018)
   3.0235
working on: 11 (name: 23050)
   3.0844
working on: 12 (name: 28083)
   3.0348
working on: 13 (name: 29030)
   2.9397
working on: 14 (name: 3063)
   2.9799
working on: 15 (name: 33044)
   2.9924
working on: 16 (name: 35028)
   2.9411
working on: 17 (name: 35049)
   2.9531
working on: 18 (name: 36046)
   2.9896
working on: 19 (name: 41006)
   2.9424
working on: 20 (name: 41029)
   2.9887
working on: 21 (name: 41085)
   2.9453
working on: 22 (name: 41096)
   2.9770
working on: 23 (name: 43033)
   2.9860
working on: 24 (name: 43051)
   2.9378
working on: 25 (name: 45000)
   3.0845
working on: 26 (name: 48017)
   2.9587
working on: 27 (name: 48025)
   2.9258
working on: 28 (name: 49024)
   2.9390
working on: 29 (name: 5096)
   2.9510
working on: 30 (name: 51084)
   2.9200
working on: 31 (name: 6046)
   2.9569
working on: 32 (name: 61034)
   2.9241
working on: 33 (name: 64061)
   2.9700
working on: 34 (name: 65084)
   2.8999
working on: 35 (name: 69000)
   2.9650
working on: 36 (name: 69007)
   2.9353
working on: 37 (name: 69022)
   2.9674
working on: 38 (name: 70011)
   2.9810
working on: 39 (name: 70090)
   3.0652
working on: 40 (name: 71076)
   4.0467
working on: 41 (name: 71099)
   3.6899
working on: 42 (name: 77062)
   3.7786
working on: 43 (name: 78098)
   3.1600
working on: 44 (name: 79073)
   3.0895
working on: 45 (name: 80085)
   3.1611
working on: 46 (name: 80090)
   3.1038
working on: 47 (name: 8068)
   3.2530
working on: 48 (name: 81066)
   3.1394
working on: 49 (name: 81090)
   3.2471
working on: 50 (name: 81095)
   3.0341
load   50 of   5
working on parameter set: slic_256, image:  50
Average boundary recall = 0.524409 for K = 64
Average underseg error  = 0.379966 for K = 64
```

K = 256

executing: [S, time, imgVis] = slic(I, 256, 25)
working on: 1 (name: 10081)
4.6431
working on: 2 (name: 14085)
4.4785
working on: 3 (name: 14092)
4.4982
working on: 4 (name: 15011)
4.7331
working on: 5 (name: 15062)
4.6597
working on: 6 (name: 16004)
4.6372
working on: 7 (name: 16068)
4.4282
working on: 8 (name: 17067)
4.5173
working on: 9 (name: 20069)
4.6092
working on: 10 (name: 2018)
4.6371
working on: 11 (name: 23050)
4.5574
working on: 12 (name: 28083)
4.4694
working on: 13 (name: 29030)
4.4851
working on: 14 (name: 3063)
4.4159
working on: 15 (name: 33044)
4.4832
working on: 16 (name: 35028)
4.5230
working on: 17 (name: 35049)
4.5414
working on: 18 (name: 36046)
4.4994
working on: 19 (name: 41006)
4.4411
working on: 20 (name: 41029)
4.4719
working on: 21 (name: 41085)
4.5129
working on: 22 (name: 41096)
4.5096
working on: 23 (name: 43033)
4.5056
working on: 24 (name: 43051)
4.4446
working on: 25 (name: 45000)
4.7255
working on: 26 (name: 48017)
4.6819
working on: 27 (name: 48025)
4.6216
working on: 28 (name: 49024)
4.6541
working on: 29 (name: 5096)
4.6232
working on: 30 (name: 51084)
4.5439
working on: 31 (name: 6046)
5.1200
working on: 32 (name: 61034)
5.2644
working on: 33 (name: 64061)
4.8491
working on: 34 (name: 65084)
4.6786
working on: 35 (name: 69000)
4.6628
working on: 36 (name: 69007)
4.5313
working on: 37 (name: 69022)
4.5891
working on: 38 (name: 70011)
4.5551
working on: 39 (name: 70090)
4.5180
working on: 40 (name: 71076)
4.5422
working on: 41 (name: 71099)
4.4864
working on: 42 (name: 77062)
4.5564
working on: 43 (name: 78098)
4.5887
working on: 44 (name: 79073)
4.5145
working on: 45 (name: 80085)
4.5776
working on: 46 (name: 80090)
4.6496
working on: 47 (name: 8068)
4.5510
working on: 48 (name: 81066)
4.5573
working on: 49 (name: 81090)
4.4960
working on: 50 (name: 81095)
4.6328
load 50 of 5
working on parameter set: slic_256, image: 50
Average boundary recall = 0.838504 for K = 256
Average underseg error = 0.258132 for K = 256

K = 1024

executing [S, time, imgVis] = slic(I, 1024, 25);
working on: 1 (name: 10081)
6.8155
working on: 2 (name: 14085)
6.6929
working on: 3 (name: 14092)
6.6256
working on: 4 (name: 15011)
6.6174
working on: 5 (name: 15062)
6.5761
working on: 6 (name: 16004)
6.4735
working on: 7 (name: 16068)
7.0616
working on: 8 (name: 17067)
7.4089
working on: 9 (name: 20069)
6.5714
working on: 10 (name: 2018)
6.6485
working on: 11 (name: 23050)
6.5926
working on: 12 (name: 28083)
6.6737
working on: 13 (name: 29030)
6.6898
working on: 14 (name: 3063)
6.7428
working on: 15 (name: 33044)
6.8505
working on: 16 (name: 35028)
6.7836
working on: 17 (name: 35049)
6.5345
working on: 18 (name: 36046)
6.5903
working on: 19 (name: 41006)
6.6839
working on: 20 (name: 41029)
6.6296
working on: 21 (name: 41085)
6.6310
working on: 22 (name: 41096)
6.6521
working on: 23 (name: 43033)
6.6137
working on: 24 (name: 43051)
6.5053
working on: 25 (name: 45000)
6.8093
working on: 26 (name: 48017)
6.9034
working on: 27 (name: 48025)
6.8488
working on: 28 (name: 49024)
6.5457
working on: 29 (name: 5096)
6.5438
working on: 30 (name: 51084)
6.5158
working on: 31 (name: 6046)
6.6111
working on: 32 (name: 61034)
6.4900
working on: 33 (name: 64061)
6.5630
working on: 34 (name: 65084)
6.5519
working on: 35 (name: 69000)
6.6214
working on: 36 (name: 69007)
6.5426
working on: 37 (name: 69022)
6.5856
working on: 38 (name: 70011)
6.5660
working on: 39 (name: 70090)
6.9460
working on: 40 (name: 71076)
6.5583
working on: 41 (name: 71099)
6.5805
working on: 42 (name: 77062)
6.6178
working on: 43 (name: 78098)
6.7155
working on: 44 (name: 79073)
6.6591
working on: 45 (name: 80085)
6.6475
working on: 46 (name: 80090)
6.5674
working on: 47 (name: 8068)
6.5326
working on: 48 (name: 81066)
6.5328
working on: 49 (name: 81090)
6.4993
working on: 50 (name: 81095)
6.5460
load 50 of 50
working on parameter set: slic_256, image: 50
Average boundary recall = 0.942185 for K = 1024
Average underseg error = 0.213435 for K = 1024

PART 1

Graph Cut Segmentation

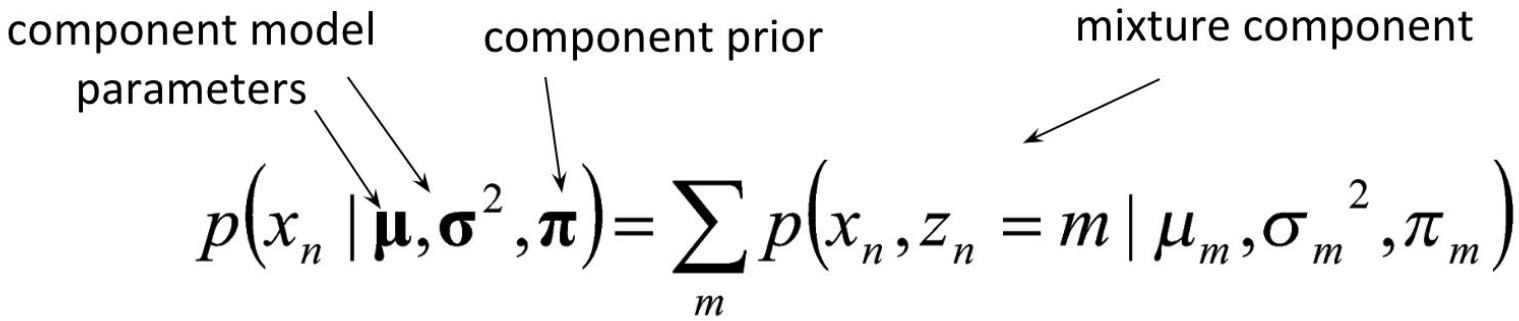
(a) Explain your foreground and background likelihood function.

We have used Gaussian Mixture Models to represent our foreground and background estimations or models.

The MATLAB function fitgmdist() was used to create the models.

The background \ foreground elements were stored in a N x 3 vector where N = number of elements in a model (BG \ FG) and 3 for each of R, G and B components of every element in the model.

The equation for a Gaussian Mixture Model is given below:



(b) Write unary and pairwise term as well as the whole energy function, in math expressions.

① Unary term:

Given a Gaussian Mixture Model for foreground and background,

$$\text{Unary}(n) = -\text{Log}\left[\frac{P(f(n) \mid \Theta_{\text{foreground}})}{P(f(n) \mid \Theta_{\text{background}})}\right]$$

n = for every nth element or pixel

f(n) = bin value or label for a pixel n.

Θ = label assigned.

② Pairwise term:

It is the sum of smoothness cost and the contrast sensitive weight.

$$\text{Smoothness cost}(x, y) = \begin{cases} 0, & x = y \\ k, & x \neq y \end{cases}$$

$$\text{Contrast sensitive term} = e^{\frac{-(g_x^2 + g_y^2)^{1/2}}{2\sigma^2}}$$

k = some constant

g_x, g_y = gradients along X and Y

σ^2 = variance.

Pairwise term = Smoothness cost + contrast sensitive term.

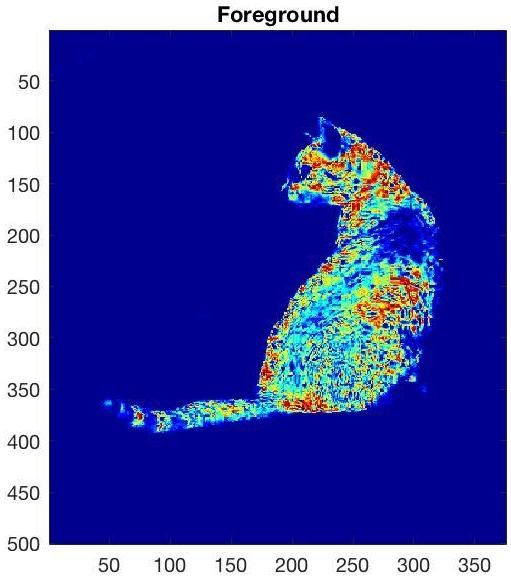
③ Energy function:

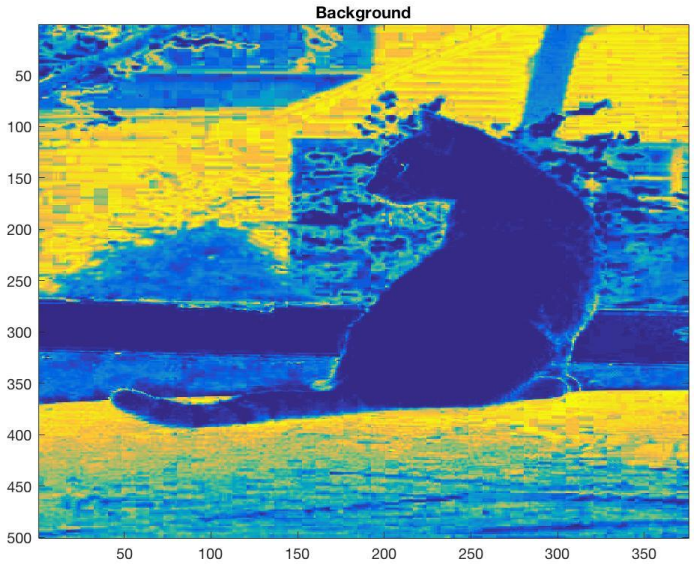
$$\hat{E}(f) = \sum_{p \in \mathcal{P}} D_p(f_p) + \sum_{p,q \in \mathcal{P}} V(f_p, f_q)$$

$D_p(f_p)$ = Unary Potential or cost of associating a label to a pixel.

$V(f_p - f_q)$ = Pairwise Potential or cost of segmenting like labels under same segment.

(c) Your foreground and background likelihood map. Display P(foreground|pixel)P(foreground|pixel) as an intensity map (bright = confident foreground).





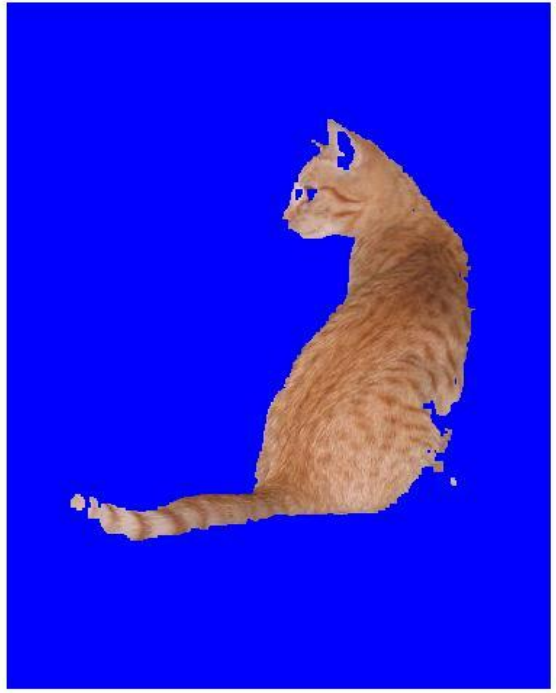
Foreground map: Bright = closer to foreground, Dark = farther from foreground

Background map: Bright = closer to background, Dark = farther from background

(d) Final segmentation. Create an image for which the background pixels are blue, and the foreground pixels have the color of the input image.

Input

Output



Graduate Credit

Modern interactive segmentation

- **up to 15 points**: Use the PolyRNN tool and annotate a small collection of images. (More details on Piazza).

Output Folder Uploaded To Given Link.

SLIC superpixel

- **up to 15 points**: Try to improve your result part 1-(e). You may try different color space (e.g., CIELab, HSV) (See Sec 4.5 in the paper), richer image features (e.g., gradients) or any other ideas you come up with. Report the accuracy on boundary recall and under-segmentation error with K = 256. Compare the results with part 1-(e) and explain why you get better results.

Using Different Color Space and Gradients

CIELab
Average boundary recall = 0.838504 for K = 256
Average underseg error = 0.258132 for K = 256

sRGB
Average boundary recall = 0.543499 for K = 256
Average underseg error = 0.439113 for K = 256

HSV
Average boundary recall = 0.541594 for K = 256
Average underseg error = 0.435412 for K = 256

Gradients
Average boundary recall = 0.539447 for K = 256
Average underseg error = 0.441484 for K = 256

I tried using different color spaces, out of which the CIELab color space gave the best result with the lowest underseg error.

For using gradients, I took the gradients of the image in the X and Y direction, after converting it to a grayscale image. However, the result wasnt an improvement, Another possibility could be to take gradients in all the color dimensions which would be an interesting case for observation.

Graph-cut Segmentation

- **up to 15 points**: Try your Graph-cut implementation on two more segmentation results. Specify the foreground object of interest with a bounding box and use Graph-cut to refine the boundary. Show us (1) the input image, (2) the initial bounding box and (3) the segmentation result.

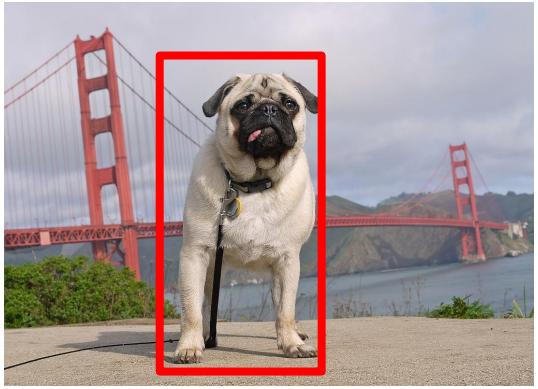




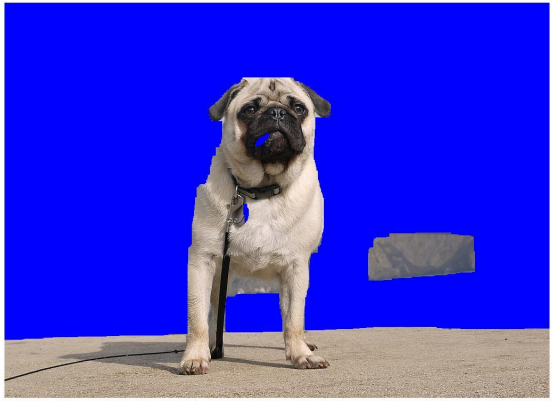

1st Try

Input

Foreground Bounding Box



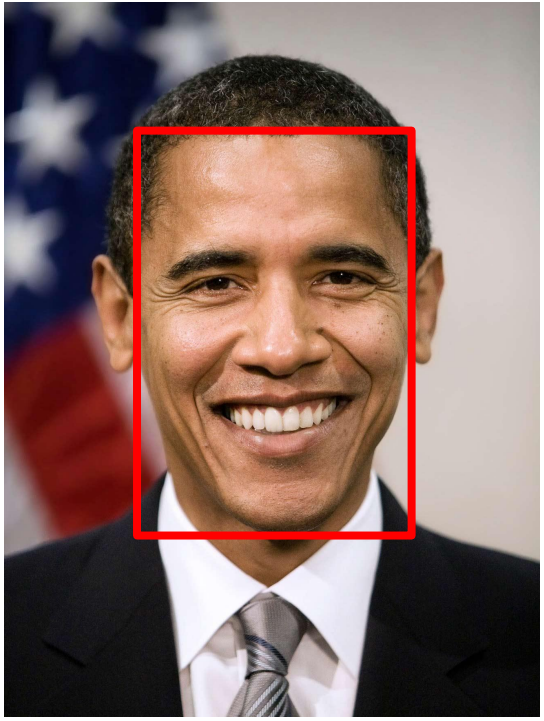
Output



2nd Try

Input

Foreground Bounding Box

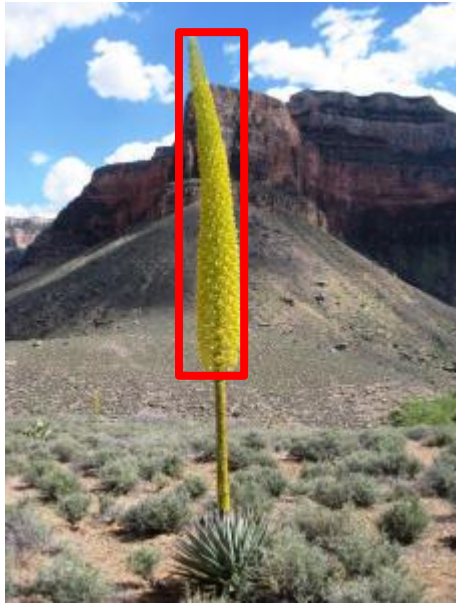


Output

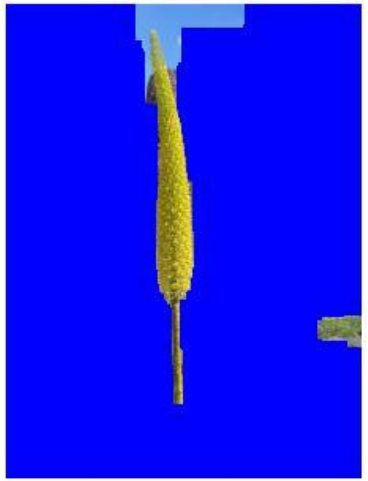
3rd Try

Input

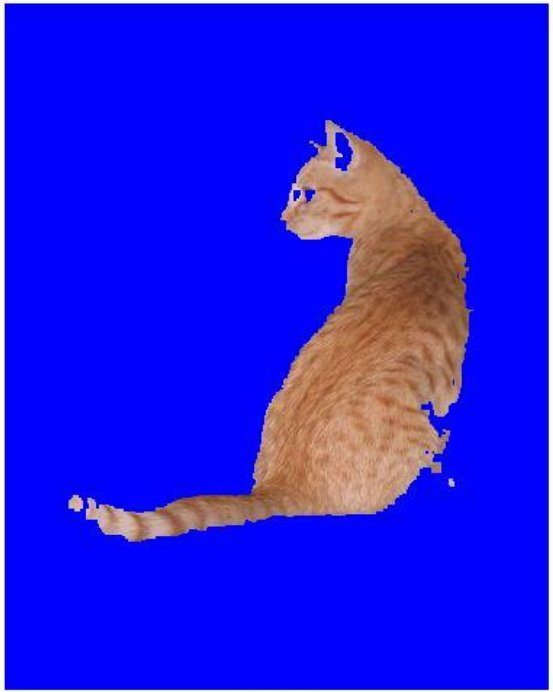
Foreground Bounding Box



Output



- **up to 10 points**: Use your Graph-cut to segment out an object in one image. Then, copy and paste the segmentation onto another image. See here for some ideas.



End.