

GrabCut Implementation

- Anmol Agarwal

Directory structure:

IPYNB NOTEBOOKS

- `Assignment3.ipynb`: Has the content corresponding to deliverable 1 (the good and bad examples and ALSO the examples which included user refinement heatmaps)
- `Experiments_<parameter name>.ipynb`: have the images/results used in the experiments which have been used to derive the inferences present in the REPORT content below

JSON and other data

- `ans_data.json`: contains the scores for all images across 10 iterations (with default parameters) and was used to make the table in deliverable 1
- `DIRECTORY ans_images`: contains directories for each image and stores snapshots of the GrabCut pipeline on each image across iterations

Basic stages and my understanding of the Grabcut algorithm

(I make references to the variables/notation used in this section in the comments in my code)

- All the pixels in the image are initialised as a part of one of the categories in the Trimap $T = \{ T_F, T_B, T_U \}$ where:
 - T_F = pixels which the user has indicated as being surely Foreground and whose foreground/background status cannot change in the future (unless the user intervenes using the REFINE tool)
 - T_B = pixels which the user has indicated as being surely Background and whose foreground/background status cannot change in the future (unless the user intervenes using the REFINE tool)
 - T_U = pixels whose foreground/background status is still not surely decided and whose status can change based on the likelihoods calculated using the GMMs
- Variables involved:**
 - Image can be considered as an array
 - $z = \{ z_1, z_2, \dots, z_N \}$ where N is the total number of pixels and in our case, $Z_i = \{\text{red channel component, green channel component, blue channel component}\}$
 - The current status (foreground/background) estimated for the pixel (X):
 - $x = \{ x_1, x_2, \dots, x_N \}$ where:
 - if $x_i == 0$: pixel is currently estimated to be a part of the background
 - if $x_i == 1$: pixel is currently estimated to be a part of the foreground
 - Estimated distributions of the foreground and background pixels
 - Here, each class (fg and bg) are modelled separately with each K full-covariance Gaussian Mixture Models (authors suggest $K = 5$).
 - In order to deal with the GMM tractably, in the optimization framework, an additional vector $k = \{ k_1, \dots, k_N \}$ is introduced, with $k_i \in \{ 1, \dots, K \}$, assigning, to each pixel, a unique GMM component, one component either from the foreground or the background model, depending on what is the current estimated category of the pixel (ie $x_i == 0$ or $x_i == 1$).
 - The variable w is used to denote the parameters of the Gaussian Mixture Models.
- An energy function E is defined so that its minimum should correspond to a good segmentation, in the sense that it is
 - guided both by the given foreground and background intensity GMMs
 - opacity (ie x) is “coherent”, reflecting a tendency to solidity of objects.
- This energy term is modelled as:
 - $E(x, \omega, z) = U(x, \omega, z) + V(x, z)$

Data Term

- The data term U evaluates the fit of the segmentation x to the pixel data z , given the model ω , and is defined for all pixels in T_U as:

Given its intensity value, how likely is a pixel to be foreground or background?

$$U(x, \omega, z) = \sum_{pixel \in T_U} -\log[h_B(z_i)][x_{pixel} == 0] -\log[h_F(z_i)][x_{pixel} == 1] + \sum_{pixel \in T_F \text{ or } T_B} \text{if (assigned category} \neq \text{user-defined category)} : \text{large}$$

Using a large constant in the error function is a **frugal way** of enforcing constraints in my implementation. Here, $h_{class}(pixel)$ is the highest likelihood (among all K components of the class) that the pixel belongs to the particular component.

Smoothness Term

- The smoothness term V evaluates whether pixels which are close to each other in terms of location and intensity belong to same category. For a pair of neighbouring pixels: if they belong to the same category, the penalty is zero ELSE the penalty depends on the similarity in their intensity and the extent of the closeness location-wise.

Given their intensity values, how likely are two neighboring pixels to have two labels?

$$V(x, z) = \gamma \sum_{\text{pixel 1, pixel 2 are neighbours}} [0 \text{ if they have same label else } 1] * dis(\text{pixel 1, pixel 2})^{-1} * (e^{-\beta ||z_{\text{pixel 1}} - z_{\text{pixel 2}}||^2})$$

Here,

- $\beta = (2 \times E[(z_i - z_j)^2])^{-1}$ where E is the expectation operator over all pairs of neighbouring pixels. Modelling this in this manner tends to lower the penalty for assigning different classes to the 2 pixels if their RGB values differs significantly despite them being close to each other location-wise
- $dis(\cdot)$ is the Euclidean distance of neighbouring pixels in 2D space

Problem Formulation

- The problem can now be formulated as:

$$x_{best} = arg\min_x(\min_{\omega} E[x,\omega,z])$$

We clearly see that the problem has 2 major steps:

- Segmentation using graph cuts (Requires having the foreground-background distribution)
- Foreground-background modelling using unsupervised clustering (Requires having segmentation)

This **cyclic dependency is modelled as** an ITERATIVE algorithm

My Implementation Details (GrabCut - Iterative Optimisation)

Step 1: Initialise Mixture Models based on user annotation

- The trimap labels (T_U, T_B, T_F) for the pixels are determined by using the initial bounding box drawn by the user.
- All pixels outside the box are put in T_B and all pixels inside the box are put in T_U
- Now, for initialisation, all pixels in T_B are assigned an estimated label as BG ($X_i = 0$) and all pixels in T_U are assigned an estimated label as FG ($x_i = 1$). Then, K GMMs are assigned for the FG label using pixels having $x_i == 1$ and K GMMs are assigned for the BG label using pixels having $x_i == 0$.
- Which FG nodes will be used to decide which of the K components for FG label is decided on the basis of K-means clustering.

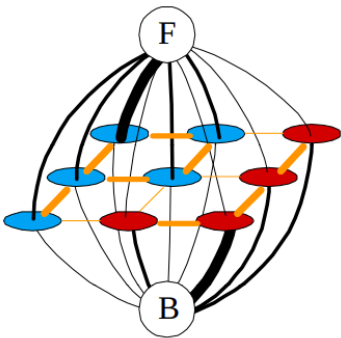
STEP 2: Assign GMM components and Learn GMM parameters

- This step is skipped in case of the 0th iteration as GMMs were already initialized in STEP 1.
- As a result of the previous iterations, due to the partitions created after applying the MIN-CUT algorithm (STEPS 3 and 4), the labels for some of the pixels might have changed.
- For each pixel in FG, we find the component among the K GMM Foreground components which has the highest likelihood of having this pixel (score_samples() function).
- Bullet 3 is repeated for pixels having BG label as well.
- Now, the GMM parameters need to be relearned. This is easily done for each of the $2K$ components by using the data points allotted to that component in BULLETS 3 and 4.

State of the Pixel	Description
pixel $\in T_B$ and $x_{\text{pixel}} = 0$	Surely background
pixel $\in T_F$ and $x_{\text{pixel}} = 1$	Surely foreground
pixel $\in T_U$ and $x_{\text{pixel}} = 0$	Possible background pixel
pixel $\in T_U$ and $x_{\text{pixel}} = 1$	Possible foreground pixel

STEP 3: Formulation of a graph and estimation of segmentation (using min-cut)

- Each pixel = node
- 2 additional nodes (source and sink) corresponding to the FG and BG classes.
- Edges :
 - Type 1: N-links** connect pixels in 4/8 neighbouring pixels around them. The weights are assigned using the smoothness term defined above. The weights remain constant as the algorithm proceeds.
 - Type 2: T-links** connect pixels to the source (FG) and sink (BG) nodes. If a pixel P is more likely to belong to FG than class BG, then the value for negative likelihood for FG will be lesser than the negative likelihood for BG and vice versa. Hence, pixel node is connected to BG using a weight derived from the negative log-likelihood of the FG class and vice versa. The weights of these edges change across the iterations.



- Energy optimization equivalent to min cut.**
- Algorithm to find the best possible labelling:** Hence, MIN-ST cut is applied to the graph to partition the graph into 2 parts such that the SOURCE and SINK belong to different parts. Due to the minimum nature of the cut, it is inferred that all nodes which are in the same partition as FG are likely to be FG pixels and vice versa. The x_i values of the pixels are modified based on which partition they lie in.
- For 2 label based segmentation, the above is solvable by standard flow algorithms and is polynomial time in theory and from my experience, it looked linear in practice.
- Repeat STEP 2

Possible STEP 4: User Refinement

The user has the option to relabel some mislabelled pixels and hence, guide segmentation in the right direction. For instance, if the user has manually labelled a pixel as belonging to BG, then the pixel is put in T_B and it's x_i value is changed to 0. The x_i value for this pixel would NOT be allowed to change in future iterations.

NOTE: EXPERIMENTS ON NEXT PAGE

Find good and bad results for deliverable 1 in assignment3.ipynb

DIRECTORY ans_images HAS my results of my implementation for all the 30 images.

Table showing deliverable 1:

- The notebook showing deliverable 1 is present in:
- Default Parameters used to construct the table:**
- gamma = 50
 - Number of components in GMM: 5
 - Connectivity mode: 8 way

Here, no user intervention was given other than the initial bounding box construction.

Across all the 30 images, following are the average values:

Number of iterations	Avg Accuracy Score	Avg Jaccard Score	Avg Dice score	Percentage of elements having Jaccard Score more than 85%	Additional details
1	0.93	0.77	0.85	53%	Best performance was obtained on doll, teddy, flower, stone2. Worst performance was obtained on: <ul style="list-style-type: none">• scissors(jaccard=0.23)• cross(jaccard=0.28)• banana1(jacc=0.43)
2	0.95	0.81	0.88	53%	As compared to 1 iteration, performance increased noticeably for: <ul style="list-style-type: none">• cross(jacc=0.33)• scissors(jacc=0.68): almost 300% increase• banana1(jacc=0.44)
5	0.96	0.83	0.90	60%	As compared to 2 iterations, following were some of the improved performances: <ul style="list-style-type: none">• scissors(jacc=0.83)• banana1(jacc=0.46)

Description of the various tweakable parameters

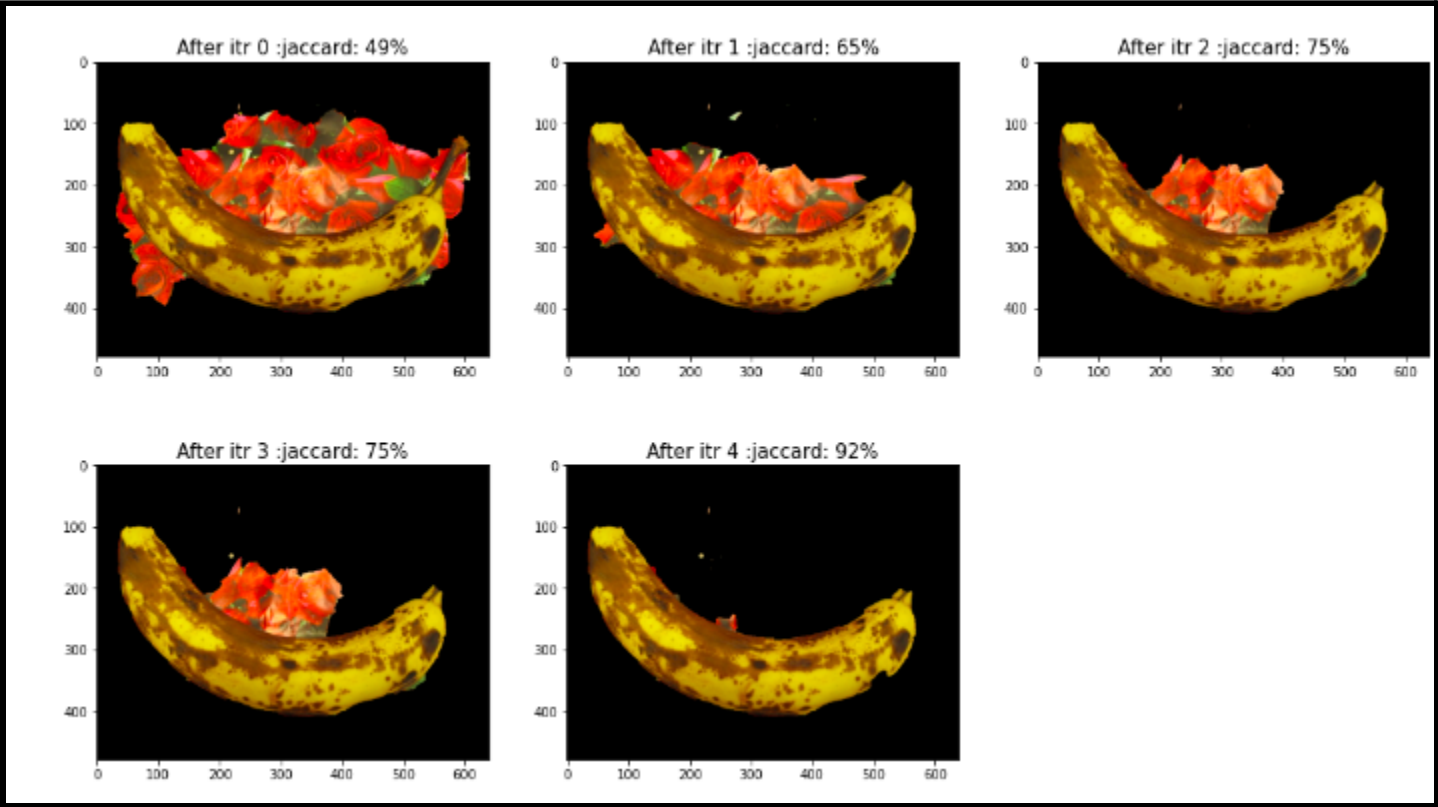
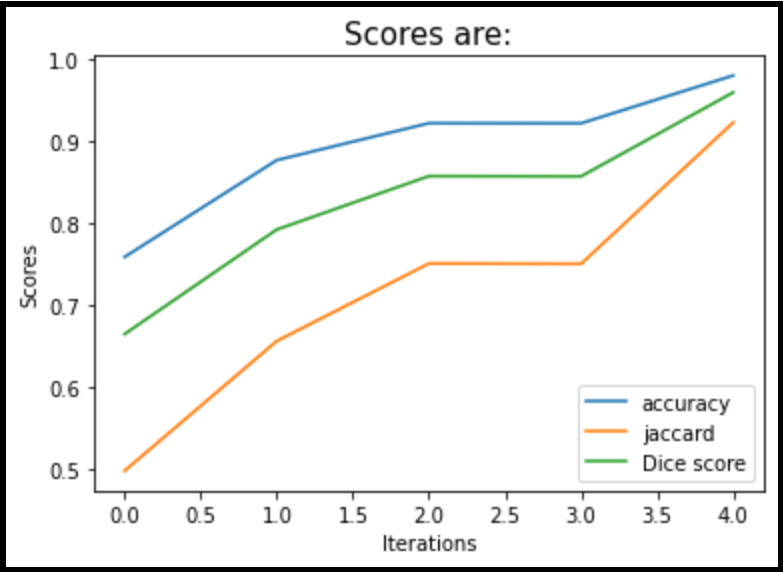
The various parameters/implementations which are tweakable are:-

Number of iterations the pipeline is performed for

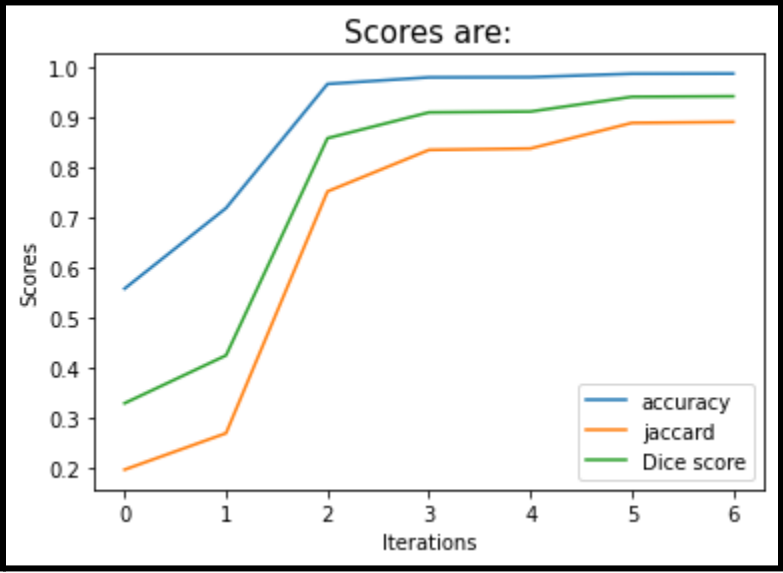
Across images, I observed that for some images, with increasing iterations, the quality of segmentation (as measured by Jaccard Score) increased while for most, it more or less remained the same. I am providing some examples for both cases:

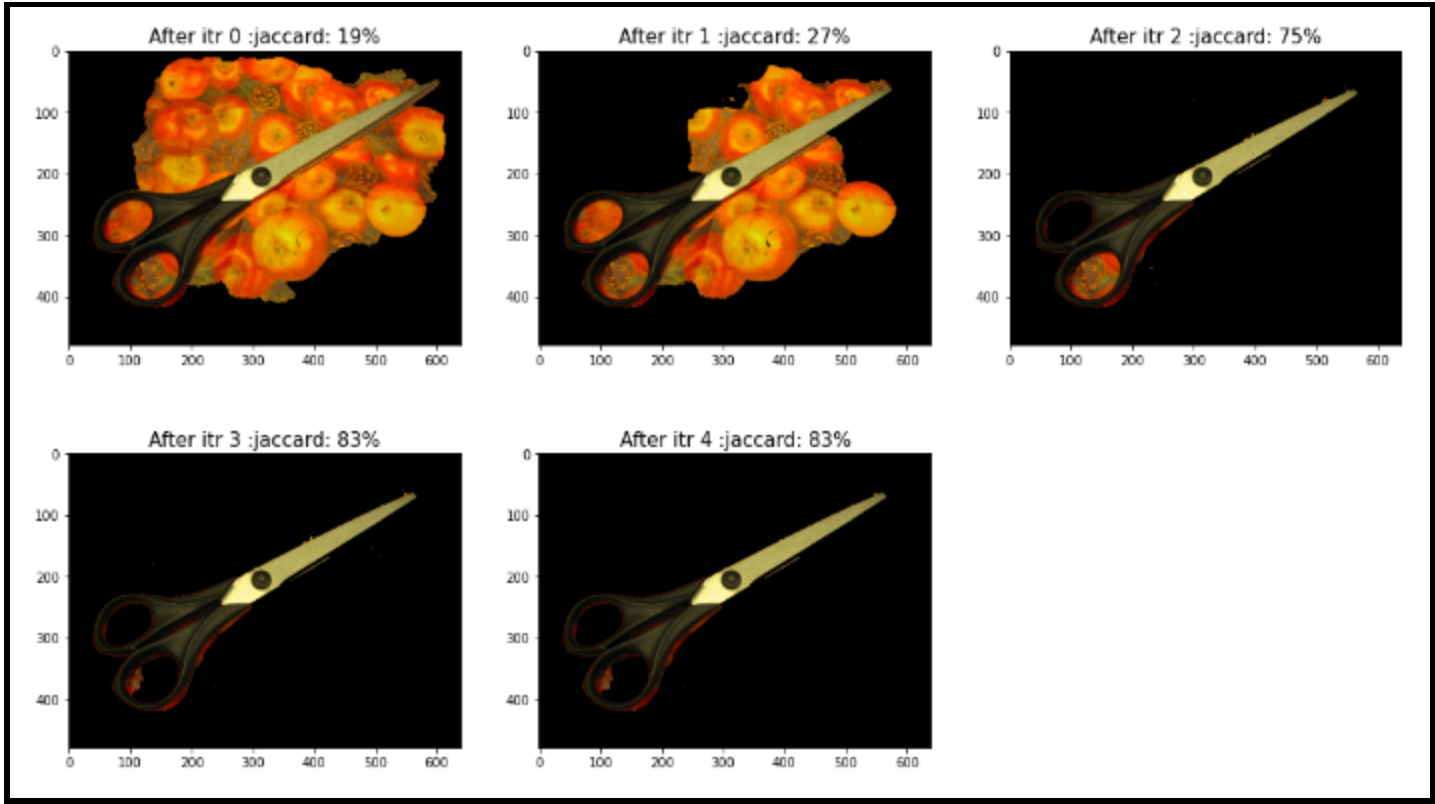
Some examples where quality increases with iterations:

Case 1: BANANA

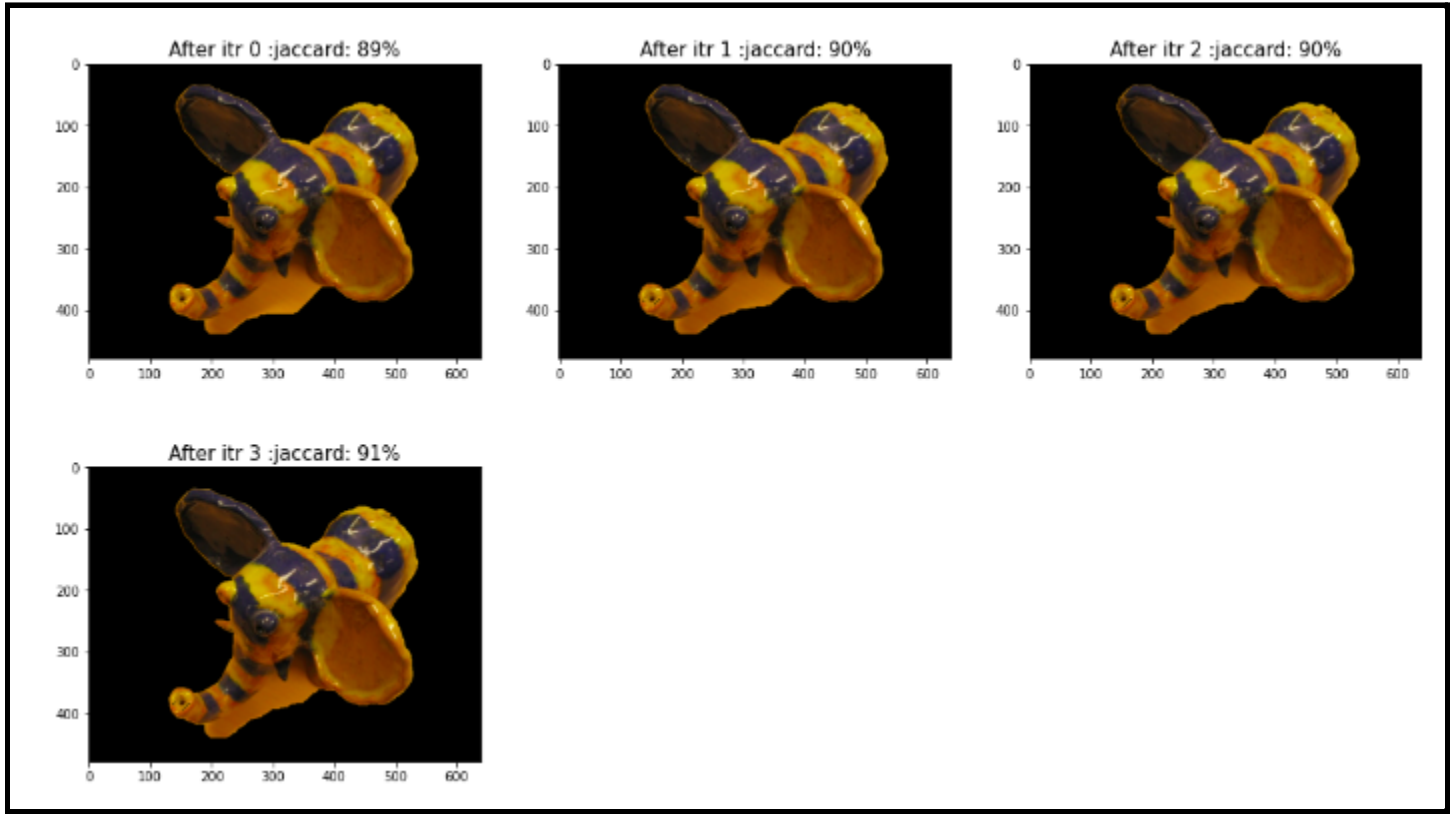
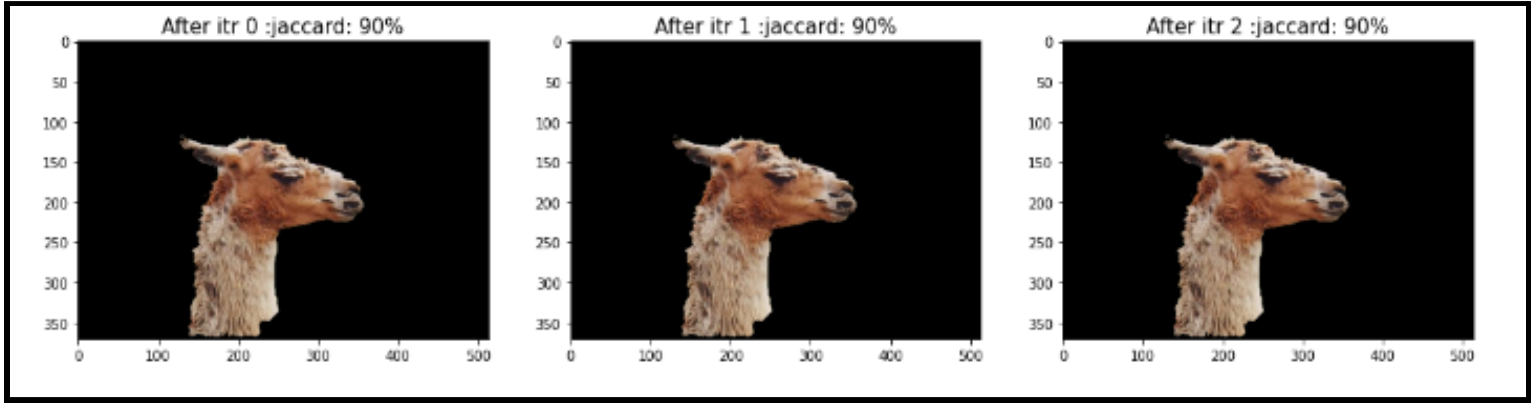


Case 2: Scissors





Examples where quality remains more or less the same (unless a human intervenes with the refining tool)



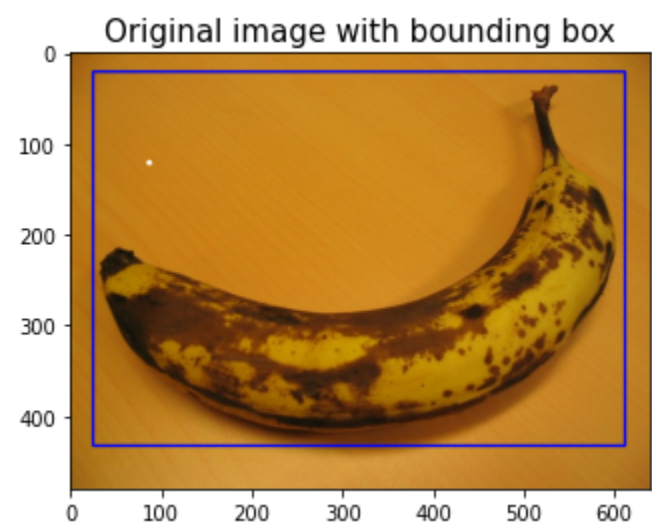
Inferences:

- In the llama example, the distribution of the foreground and background pixels is fairly obvious to the algorithm even at the beginning (AS the BOUNDING BOX of the llama INCLUDES very few ACTUAL BG pixels). The initial bounding box itself contains very FEW actual BG pixels. Hence, the algorithm is able to learn the accurate distributions for both the BG and FG classes in the very first stage itself and hence, no noticeable difference is observed after the results of the first distribution.
- For banana and scissors, a large part of the bounding box initially has pixels of the background and these **background pixels are used to model foreground GMMs initially** (since they belong to T_U). As a result, it takes the algorithm some iterations to relate these BG pixels (which were there inside the bounding box) to the BG class. This happens as:
 - In the 0th iteration, both FG and BG have GMMs corresponding to the RED texture in the 2 images. Hence, from a unary potential point of view, choosing either label for table pixels in the bounding box is equivalent as far as energy is concerned.

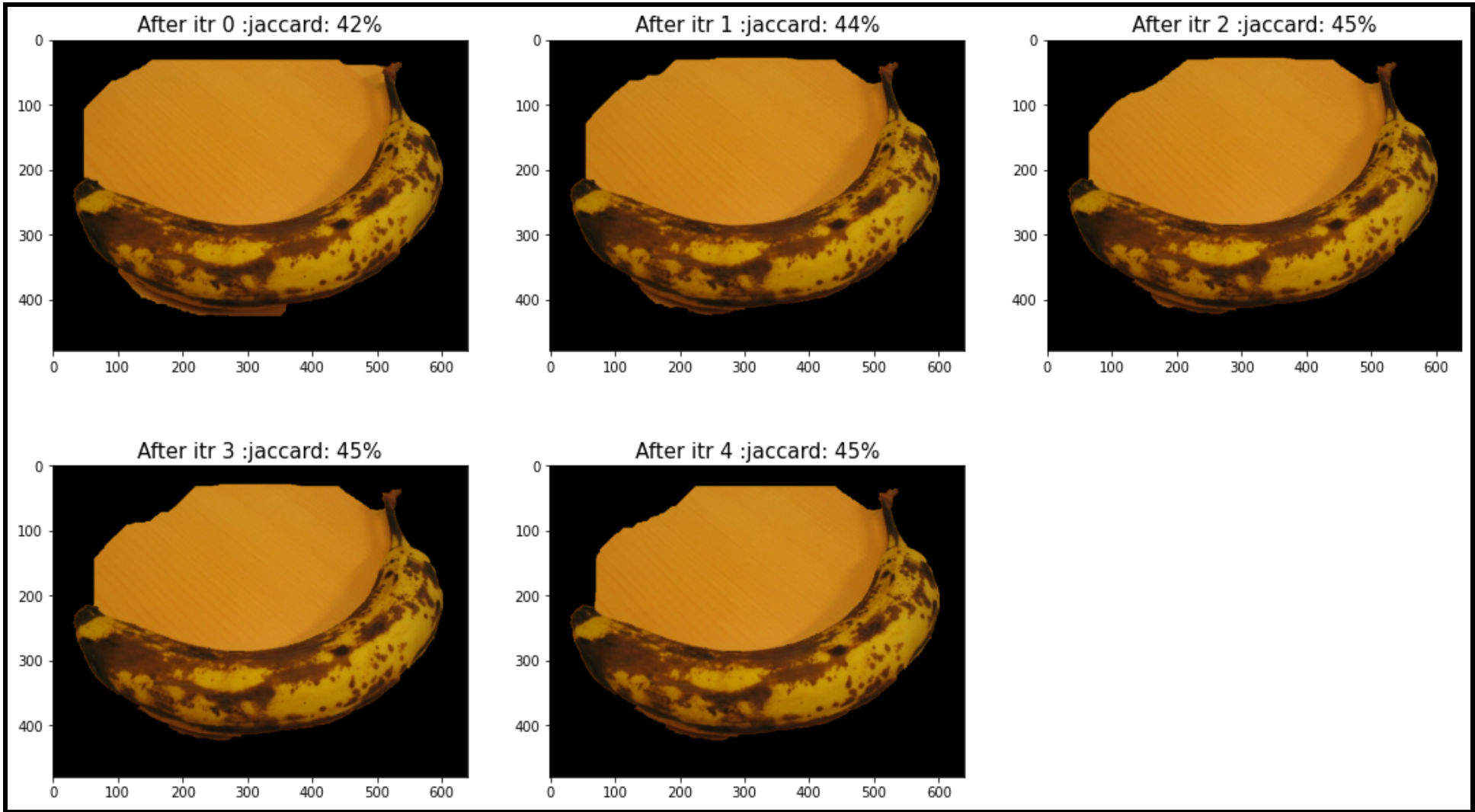
- But the edge weights (edge 1) between the RED pixels in the bounding box and RED pixels outside bounding box (which are sure to be BG as they belonged to T_B) is HIGHER than the edge weights (edge 2) between the red pixels and the scissor/banana boundary (since $||z_1 - z_2||$ will be higher in the second case). It is more costly to cut edge 1 than edge 2 and hence, the algorithm improves as soon as it learns this.

Choice of gamma (used to model the penalty if two neighboring nodes had different labels)

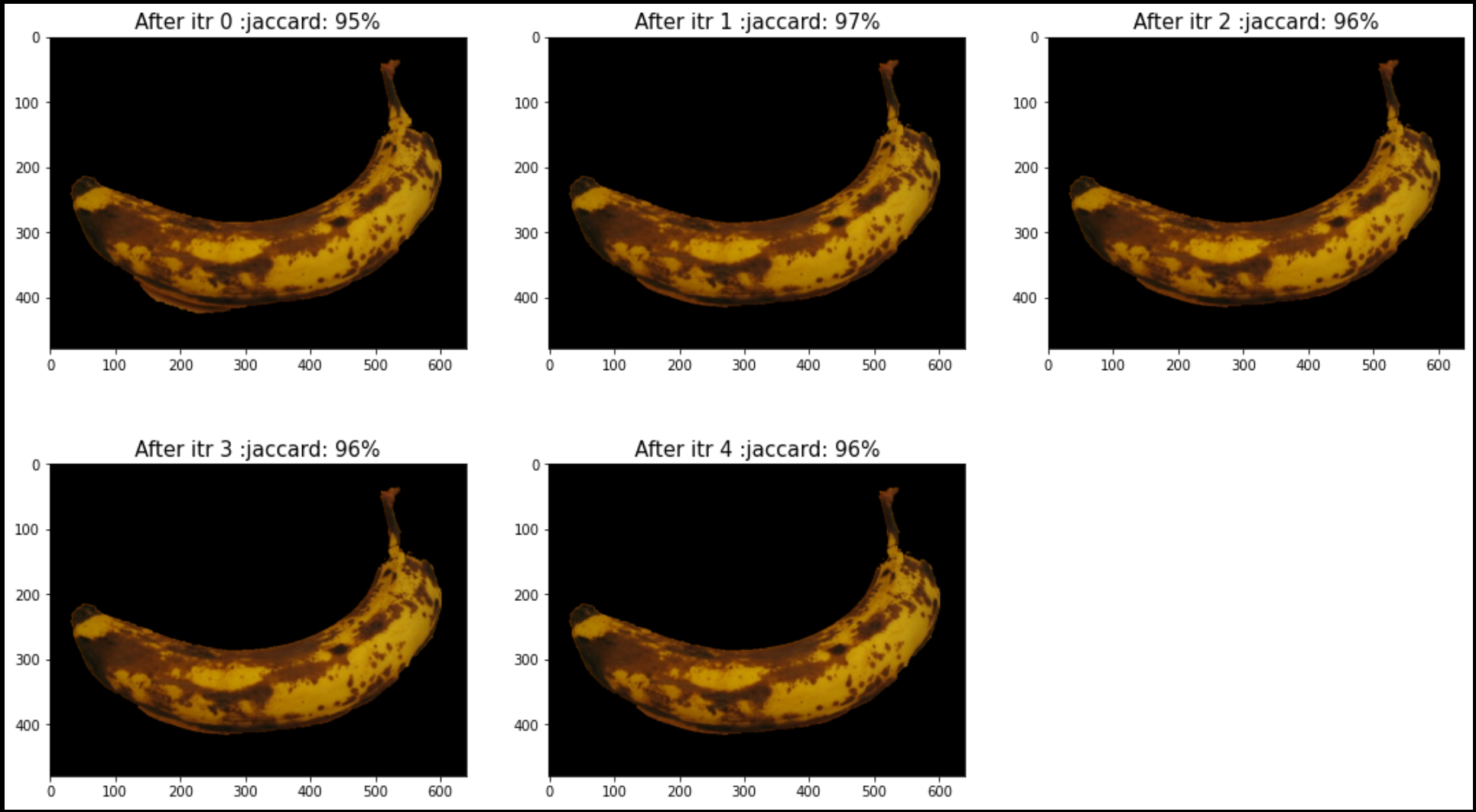
Example 1:



Case 1: gamma = 50



Case 2: Gamma = 200

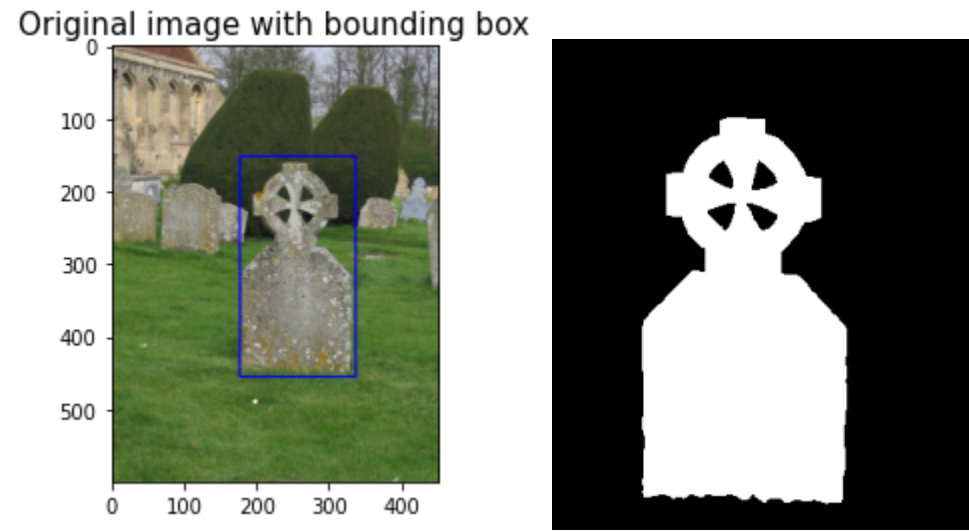


INFERENCE:

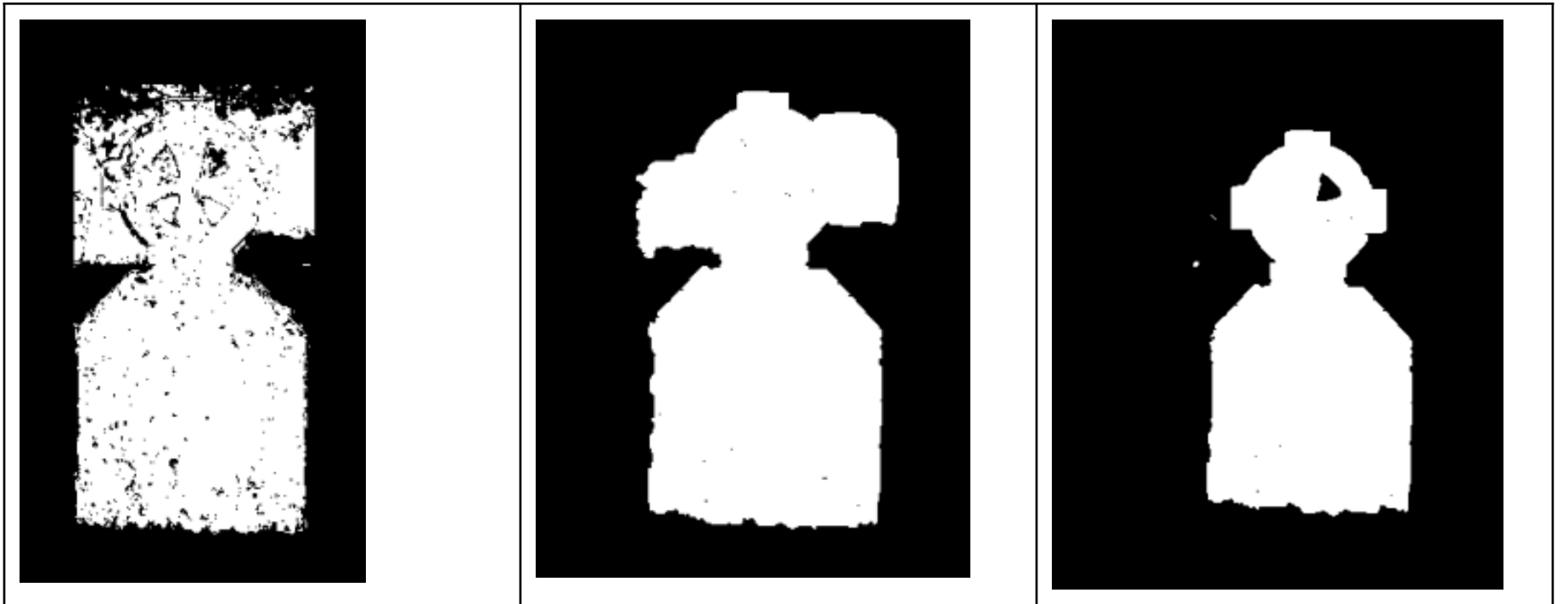
- In both cases, a part of the table was a part of some GMMs in FG and BG class. Hence, it seems that the value of gamma played the deciding role here.
- Gamma essentially weights the inter-pixel edges against the class edges. A larger gamma makes it harder for inter-pixel edges to be cut, placing more importance on continuity over the likelihood of the pixel being similar to other BG and FG pixels (modeled with the GMM distributions).
 - **GAMMA not only decides preference of inter-pixel weights VS class weights BUT it also increases the difference between INTER-PIXEL weights of 2 similar pixels and the inter-pixel weights between 2 dis-similar pixels.**
- In the first case, the BG table pixels misclassified as FG essentially had effect of 2 binary potentials:
 - The edge wt corresponding to the connections with the banana yellow pixels (WT 1)
 - The edge wt corresponding to the connections with the TABLE pixels outside the bounding box (WT 2)
- Now, due to the nature in which binary edge wts were modeled, WT 2 > WT 1 (as table pixels are more similar to each other than a table pixel is similar to a banana boundary pixel). ‘
- On further increasing the value of GAMMA, the difference between WT 1 and WT 2 increases further and at some point, it becomes enough for the table pixels (within the bounding box) to get classified as BG.

Example 2:

Image and ground truth:



Gamma: 1	Gamma: 20	Gamma: 50
----------	-----------	-----------



Inferences:

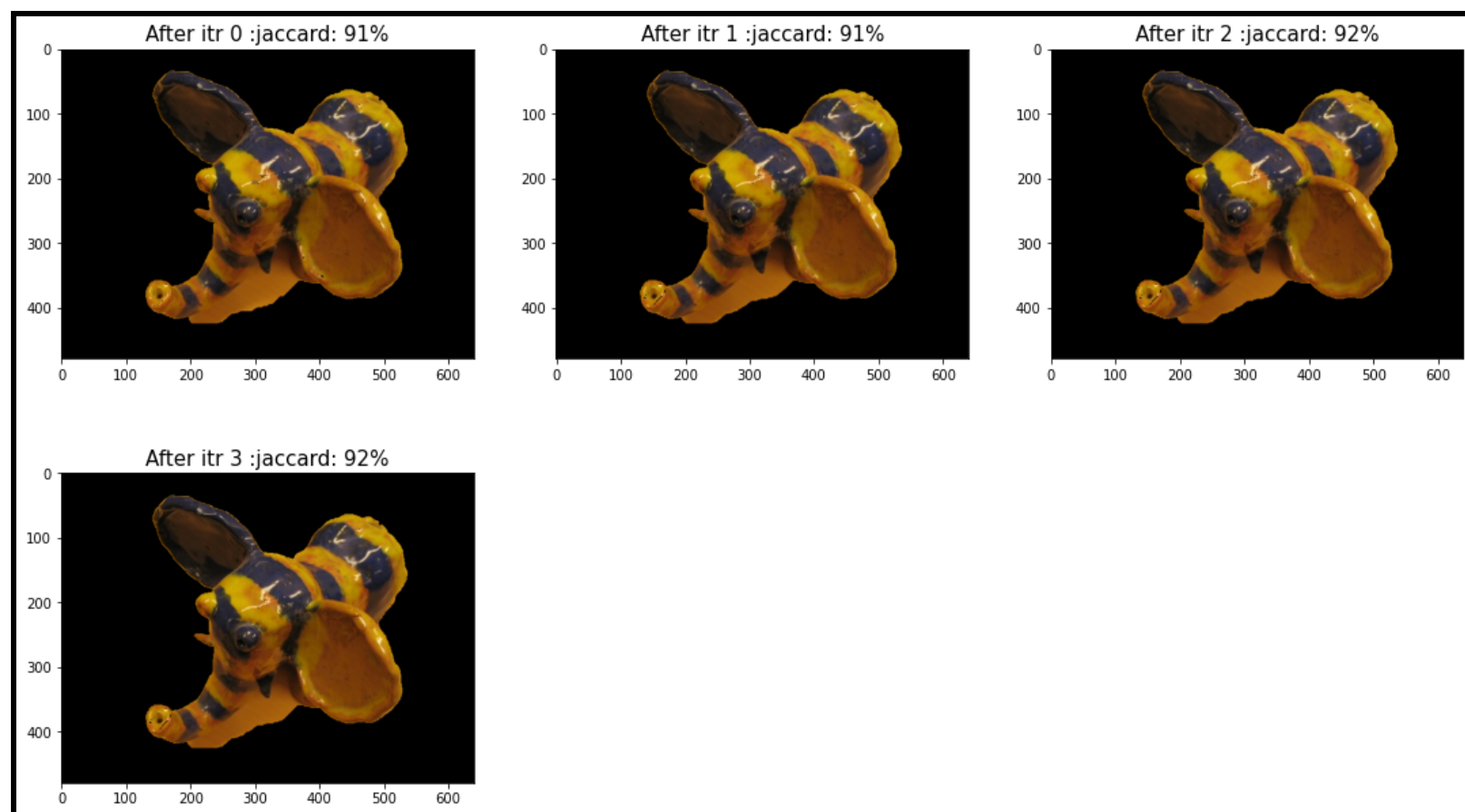
- In the first case, when $\gamma = 0$, the algorithm gives importance to class distributions and the contribution of neighboring pixel similarity to energy function is ZERO.
 - Since the Dark green bush part is present both inside and outside the bounding box and hence, both FG and BG are likely to have one GMM in their distributions allotted to model this distribution.
 - As a result, the green bush inside the bounding box does not change classes later on (since there is no INCENTIVE from an energy perspective as is in the case **WHEN GAMMA = 50**, where classifying parts of the bush with different labels will lead to severe penalty due to inter-neighboring bush pixel edge weights.)
- In the second case, we see some notion of continuity (essentially fills the black holes in the first case). We see that mostly the black holes (which were surrounded by WHITE holes on either side and hence, their misclassification would have led to cutting of 8 edges of small weights) change labels to FG.

The number of GMMs (K) per label

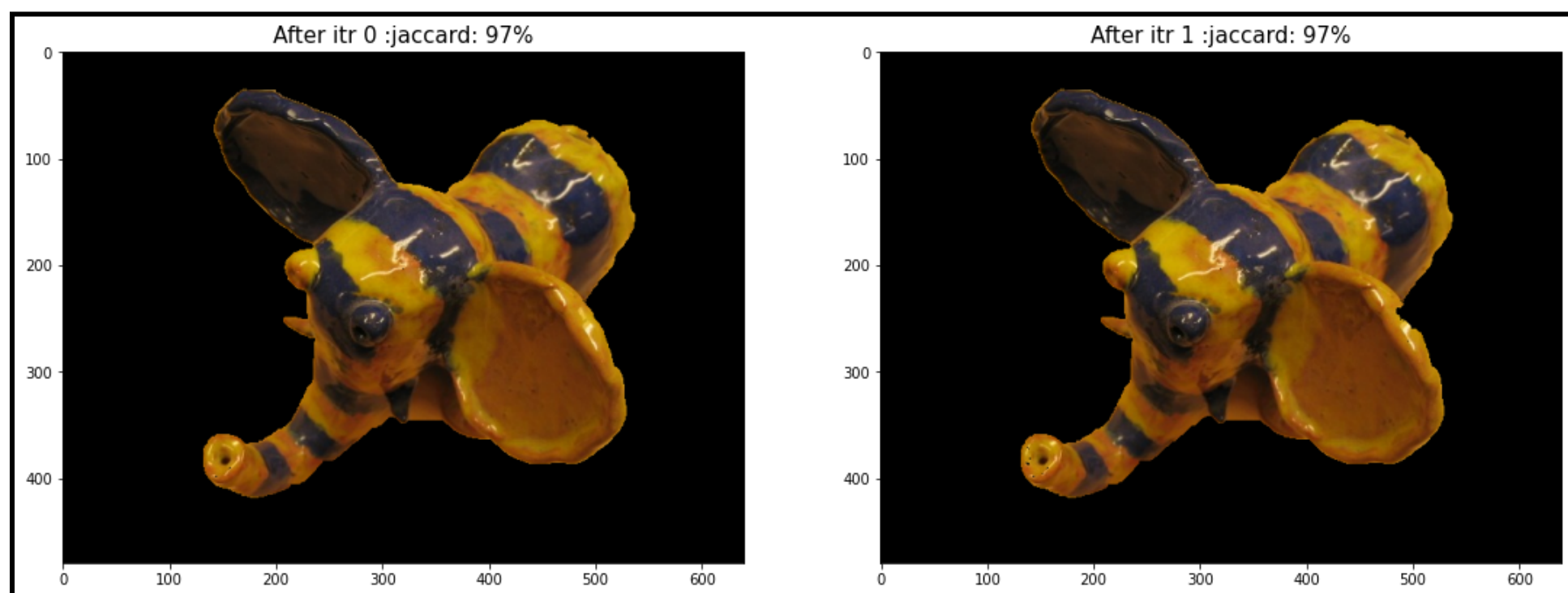
Original image:



Case 1: INPUT GMMS allowed=2 , gamma value = 50



Case 2: INPUT GMMS allowed=10



Inference/Observation:

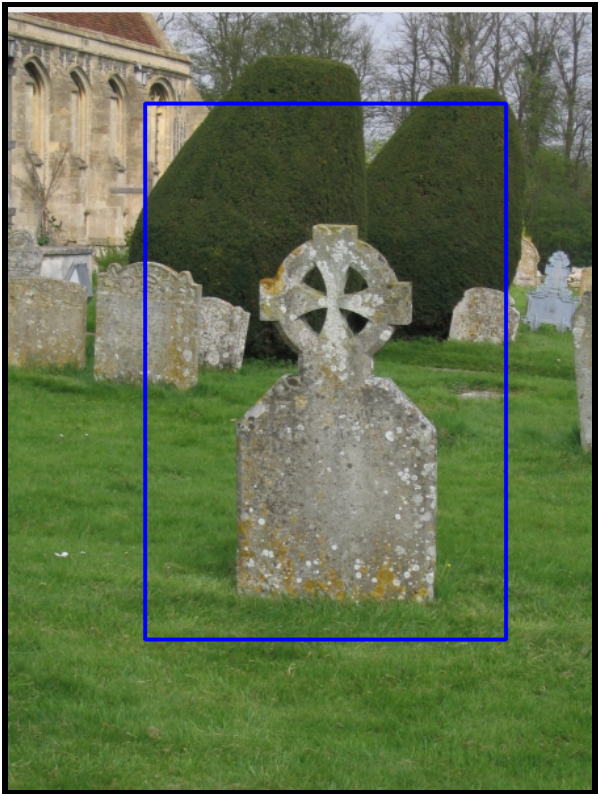
- increasing the number of components generally gives more types of textures to be captured in both the FG and BG category. It also **increases the granularity** in which these distributions can be captured. In case 1, since both FG and BG are almost yellow, there is NO ROOM to capture the LIGHTER ELEPHANT YELLOW and the DARKER SHADOW YELLOW separately. As a result, based on binary edge weights, the shadow is classified to be a part of the object (FG) itself.
- In case 2 where number of GMMS allowed = 10, I think the model was separately able to capture both the shadow dark yellow and the original object yellow in the initial BG GMMS (due to the part of the shadow outside the bounding box). Hence, it got classified as BG (since the cost of being classified as BG decreased and supported the binary potential urge to group with the other neighboring shadow pixels outside the bounding box.).
- However, it might be wrong to conclude that increasing the number of GMMS might lead to increased jaccard score. I believe the effect will be similar to overfitting a ML model. Theoretically, in the grave image,



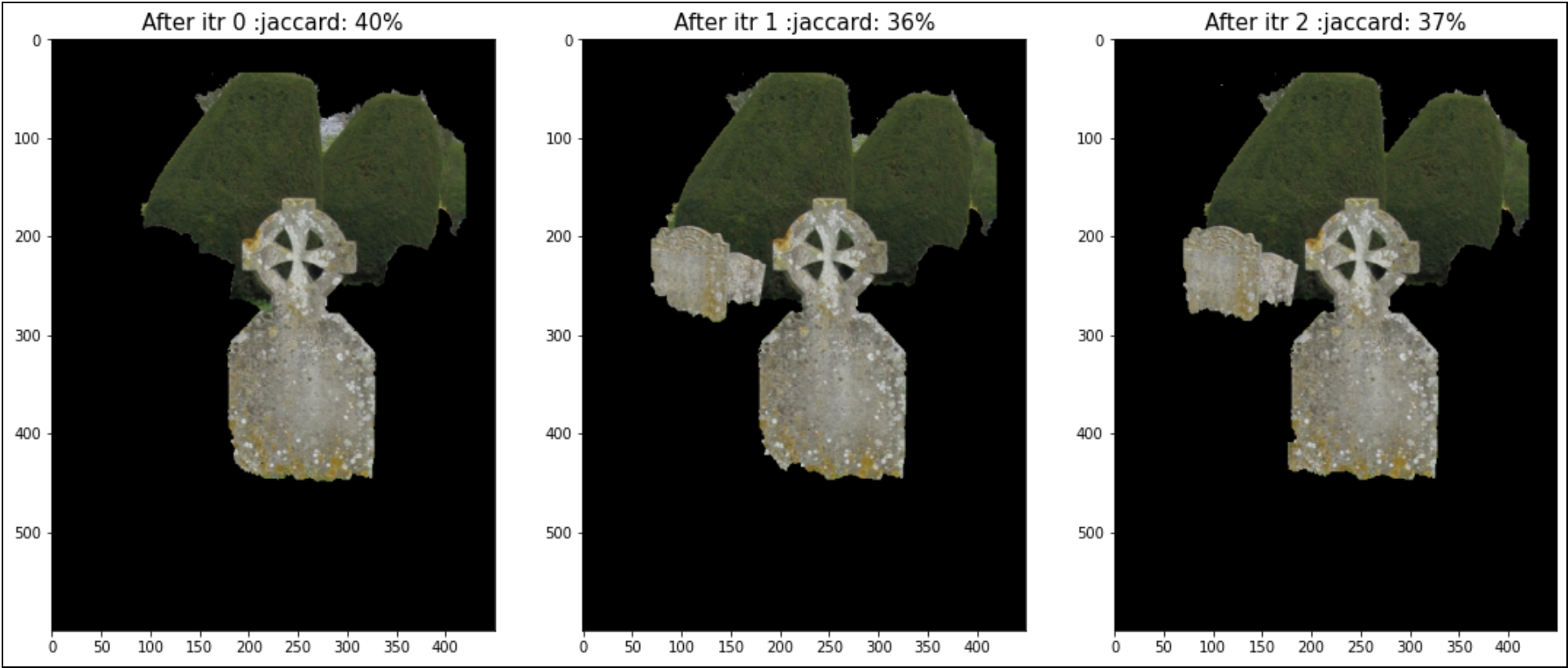
- - The shadow person of the human's leg is supposed to be classified as BG. However, if given enough GMMs, the foreground label will be able to have an entire GMM dedicated to just the shadow pixels and since the less shadow pixels will have less weighted edges with the ground pixels (since shadow ~ black and ground ~ light brown), the shadow pixels are likely to stay labeled as FG (in case theoretically many GMMs are provided).
-

Effect of tight or loose initial bounding box

Case 1: Loose bounding box



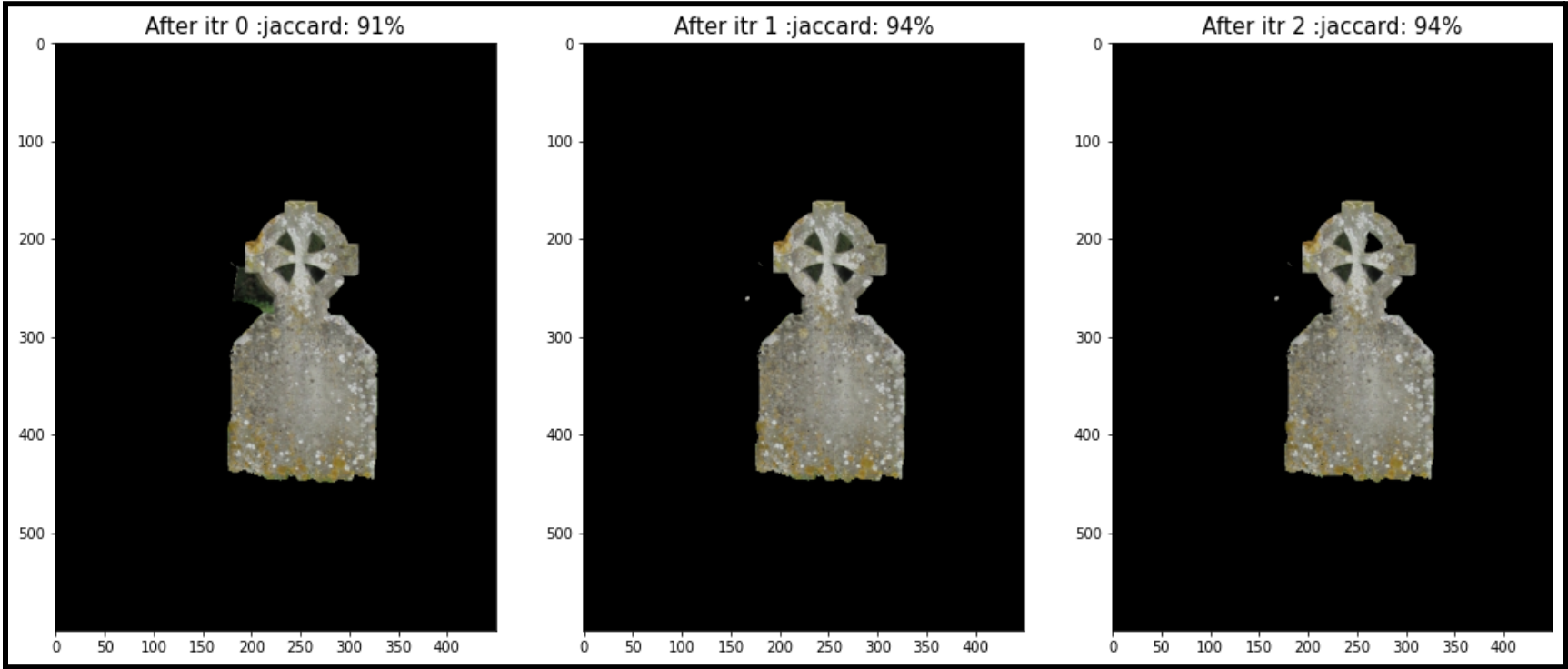
Result:



Case 2: Tight bounding box



Result:



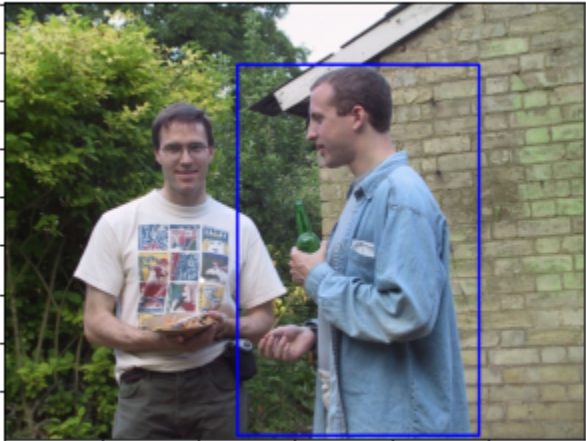



Inference/Observation:

- We notice that segmentation is better in case of the tighter bounding box.
- While change in bounding box size can affect distributions, for the above example, I believe the binary potential to be responsible as stated in the below bullet.
- The reason for the mis-classification of the DARK GREEN BUSH in the 1st case is that:
 - In both cases, some part of the GREEN BUSH is present inside and outside the box. Hence, it is fair to assume that in both cases, atleast one of the GMMs accounts for the dark green color in the bush.

- Now, since unary potential corresponding to both FG and BG would be same, the binary edge weights are likely to be the deciding factor.
- In case 2: Bush outside the box cannot be changed to FG (since they belong to F_B and are connected to node BG with a wt of INF). As a result, they compel the pixels inside the bounding box to be classified as BG as well.
- In case 1, this does not happen.

Another example:

Type	Image of Bounding Box	Result
Large bounding box	<div>Original image with bounding box</div> 	
Small bounding box	<div>Original image with bounding box</div> 	

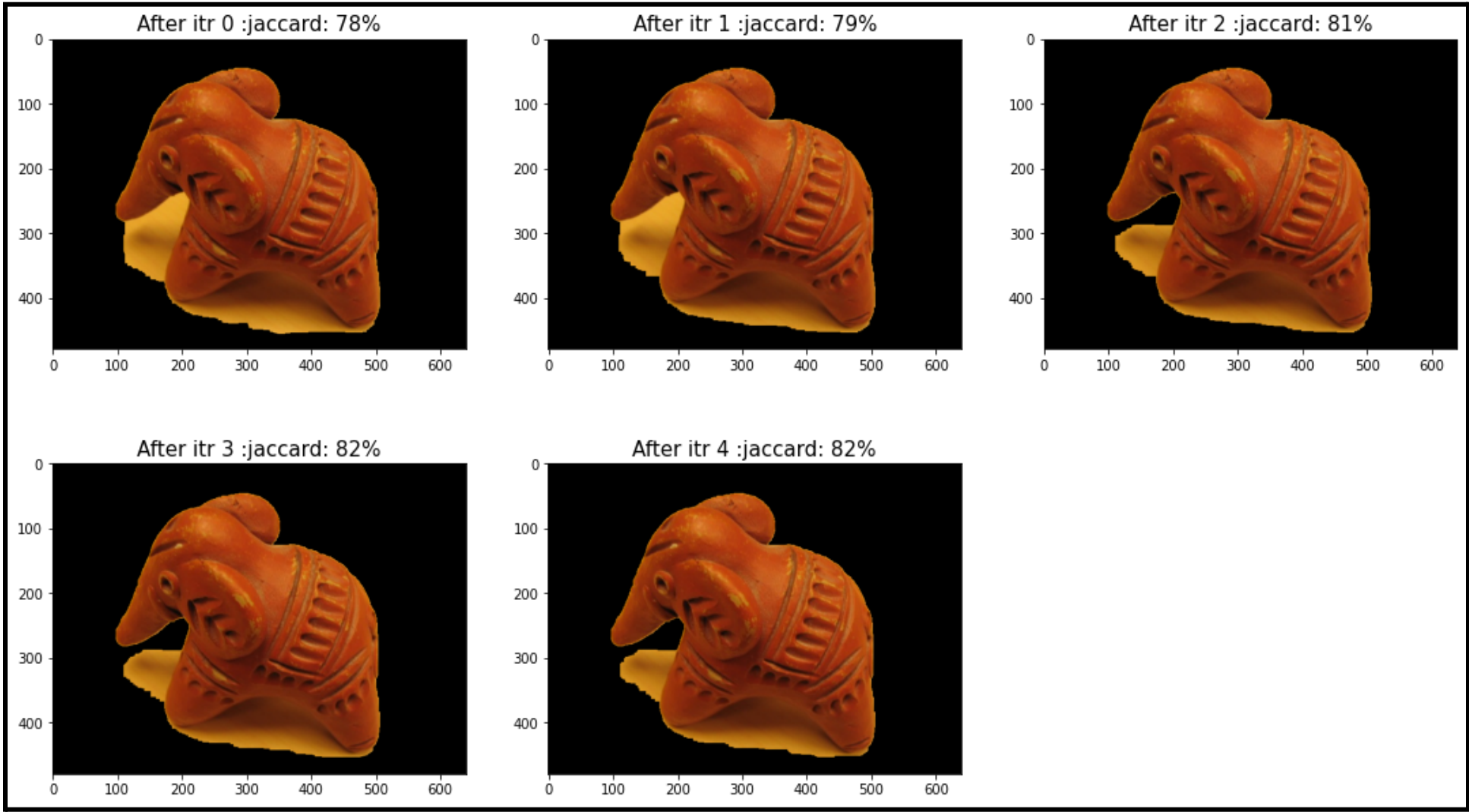
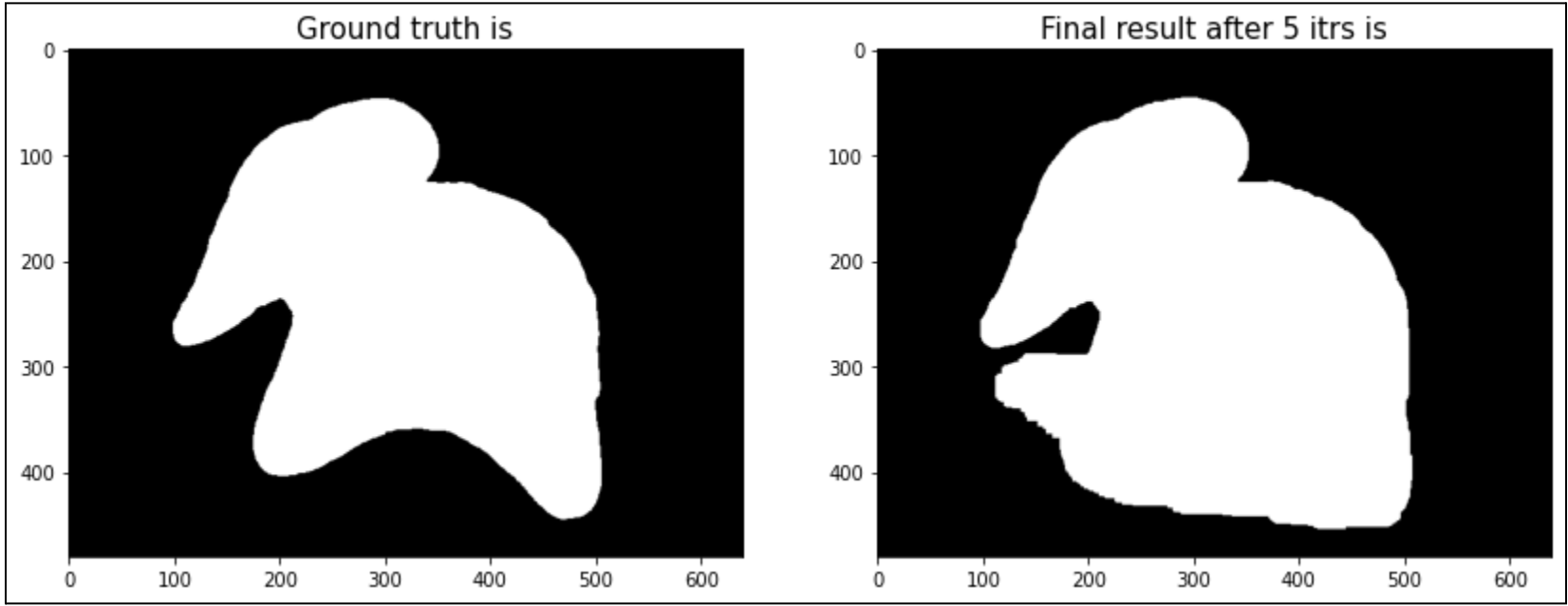
Neighborhood being defined as 4-based connectivity or 8-based connectivity

Is more the better ? Does time taken change ?

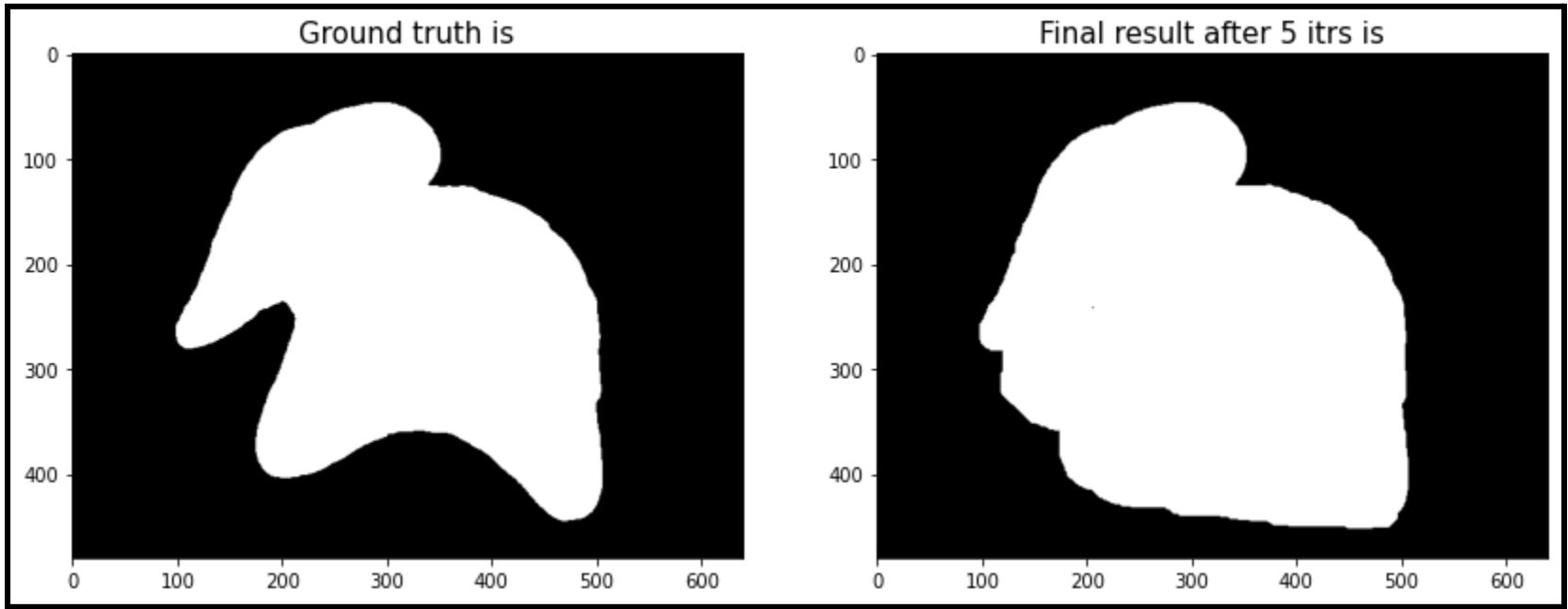
Original image:

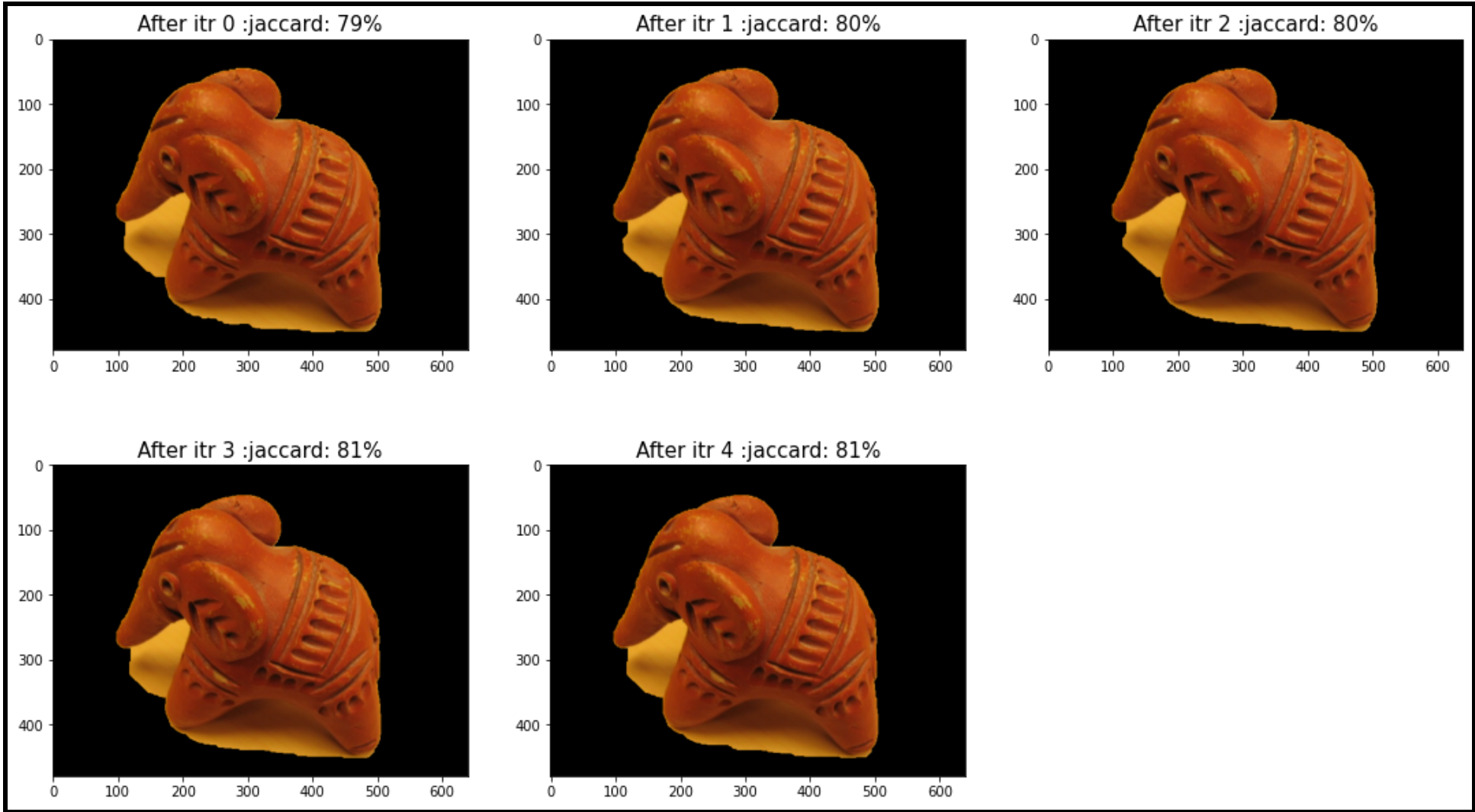


For connectivity 4:



For connectivity 8:

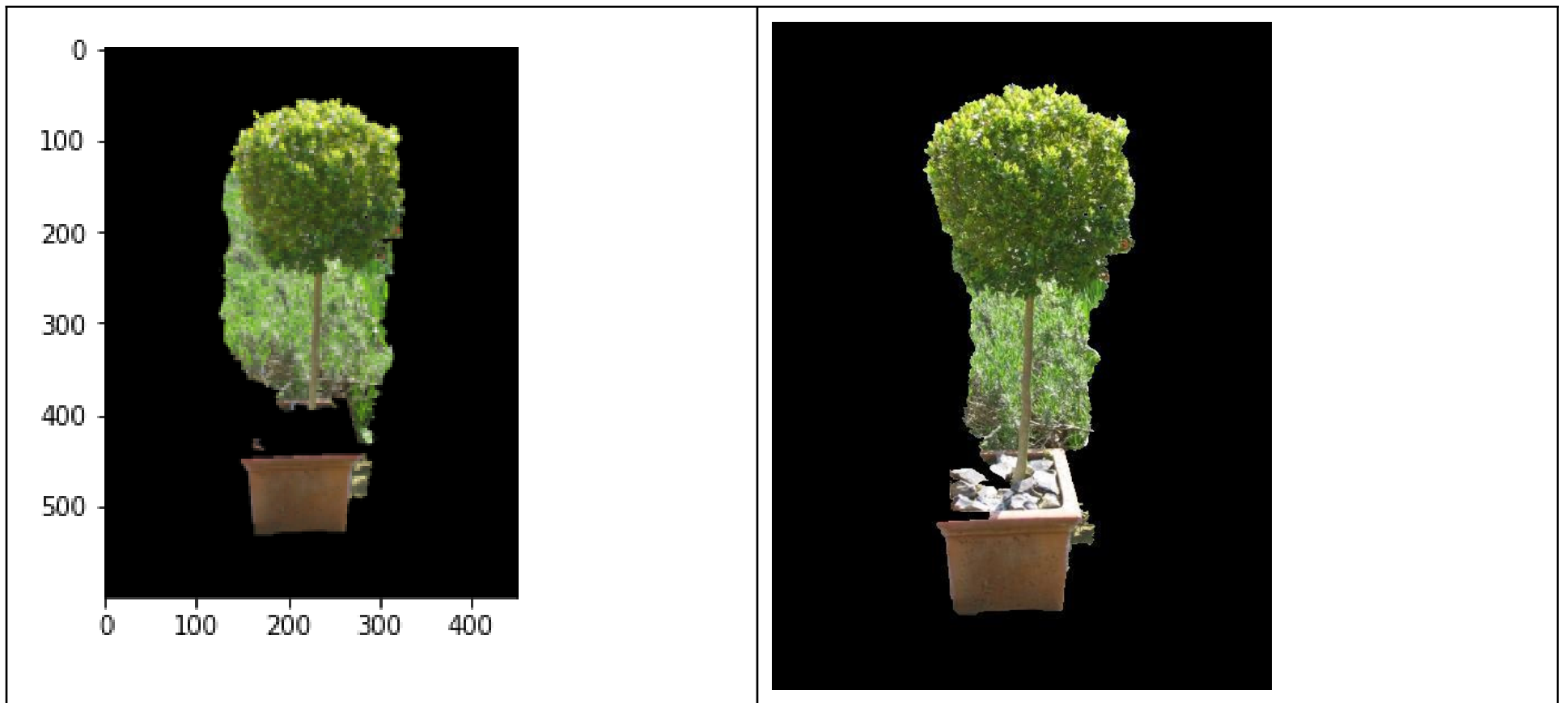




Inferences:

- My first observation (in general) was that:
 - The 8 connectivity case takes a lesser number of iterations to reach convergence in most cases.
 - However, the time taken per iteration for “8” case was higher than the “4” case.
 - As a result, the overall convergence time for challenging images was more or less the same.
 - For easier images (with less variations in pixel intensities), the “8” connectivity generally concluded quicker.
- For this specific image,
 - The edges in the “8” case are smoother whereas the edges in the “4” case are generally crooked/spiky (look at the area towards the “trunk” of the elephant).
 - Let S = table pixels in the area between elephant’s trunk and its leg
 - We see that in the 4 case, the results are better as in the 4 case, the table below the trunk of the elephant and the elephant’s leg has been correctly classified as BG (whereas it has been misclassified in the “8” stage).
 - The reason might be due to the fact that the pixels in “S” are surrounded by elephant pixels in all sides.
 - Since the number of edges around such a pixel is lesser in “4”, the cost of cutting the edges between the pixels in “S” and the pixels in (elephant trunk and leg) is lesser and hence, affordable.
 - Since the number of edges is more in “8”, the penalty for cutting 8 edges (in case the pixel in “S” is classified as BG) increases and seems to dominate, leading to prefer continuity over unary potentials.

Connectivity: 4 [after 3 iterations] (slower to converge)	Connectivity: 8 [After 3 iterations] (faster to converge)
---	---



Some references:

- “GrabCut” — Interactive Foreground Extraction using Iterated Graph Cuts (Rother et al)
- Implementing GrabCut (Talbot and Xu)
- Excellent introduction to GMMS: [Gaussian Mixture Model | Brilliant Math & Science Wiki](#)
- Slides used in CMU: <http://www.cs.cmu.edu/~16385/lectures/lecture27.pdf>

Directory structure:

IPYNB NOTEBOOKS

- Assignment3.ipynb: Has the content corresponding to deliverable 1 (the good and bad examples and ALSO the examples which included user refinement heatmaps)
- Experiments_<parameter name>: have the images/results used in the experiments which have been used to derive the inferences present in the REPORT content below

JSON and other data

- ans_data.json: contains the scores for all images across 10 iterations (with default parameters) and was used to make the table in deliverable 1
- DIRECTORY ans_images: contains directories for each image and stores snapshots of the GrabCut pipeline on each image across iterations