CV Assignment 4 Report

Pratyay Suvarnapathaki, 2020111016

Question 1

Experiments

Note:

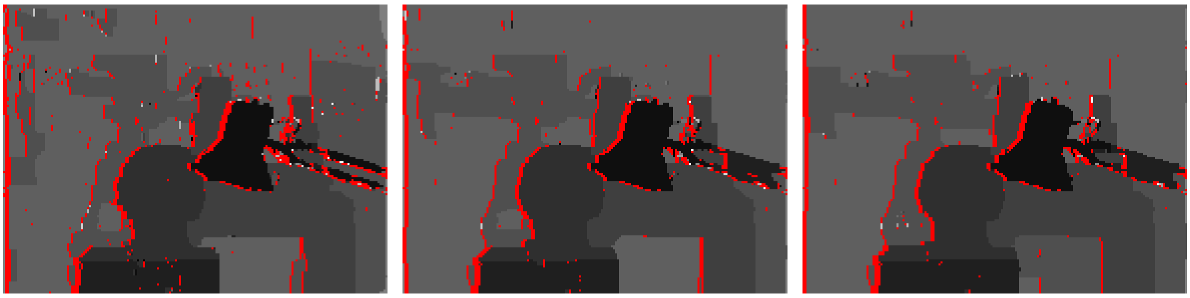
For the sake of run time (before switching to Kaggle), I've scaled the image by half. Subsequently, I use nearest neighbour interpolation in convert_disparity_to_image() to scale it back up. This has probably resulted in higher error values than if I were to do it normally.

1a. Varying Lambda

As per the algorithm, the value of Lambda is inferred from that of K.

Thus, Lambda can be varied by varying K instead.

K	Runtime	% Occluded	Total Err	Gross Err
10	59.7s	1.70	40.4	34.7
56.70 (calced)	1m16s	1.31	36.6	22.1
100	58.8s	1.21	39.5	30.6



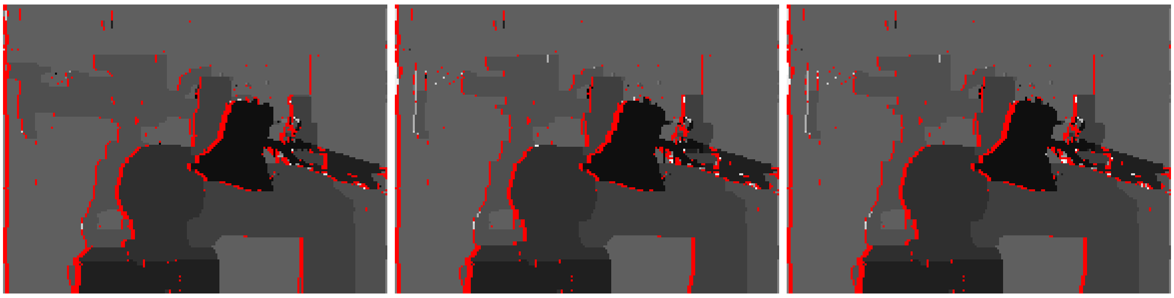
K=10 56.7 100

- K determines the occlusion cost.
- When K is low, more pixels tend to be occluded. SInce all pixels start out as occluded in my implementation, this can be termed as an 'occlusion inertia' of sorts.

- As K increases, it is more costly to label pixels as occluded, hence it is decreasingly favoured. As a result, the 'forcefully' assigned labels tend to be unreliable, as indicated by a higher gross error for K=100.
- It is important to note that the time for the default K should not be considered since there is also the factor of actually calculating the K. Largely, I would conclude time isn't impacted here as assigning a different K isn't any different in terms of computational cost.

1b. Varying Iterations

maxIter	Runtime	% Occluded	Total Err	Gross Err
1 (default)	1m16s	1.31	36.6	22.1
3	3m12s	1.24	34.5	15.37
5	3m58s	1.25	34.5	15.35

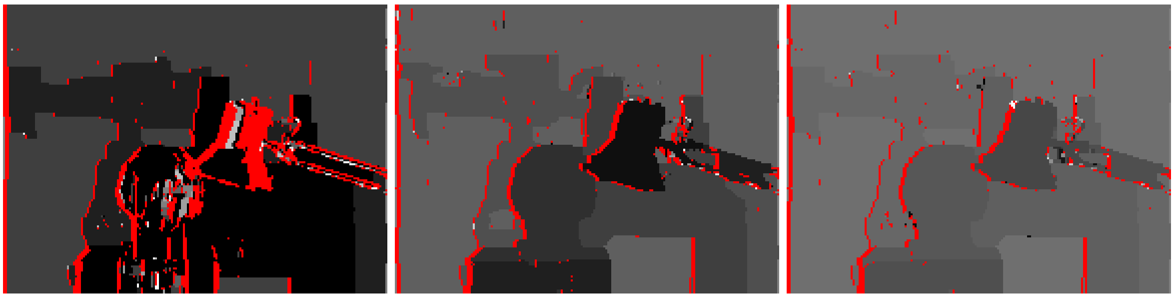


It=1 3 5

- Naturally, runtime increases with iterations.
- In terms of metrics, we see the law of diminishing returns take effect. Although some marginal improvement can be seen as we go from 1 to 3, the same cannot be said for the 3-to-5 jump.

1c. Varying Disparity Range

dispRange	Runtime	% Occluded	Total Err	Gross Err
-4 to 4	42.6s	10.77	60.6	2.99
-8 to 8 (def.)	1m16s	1.31	36.6	22.1
-16 to 16	2m25s	1.19	60.4	55.6

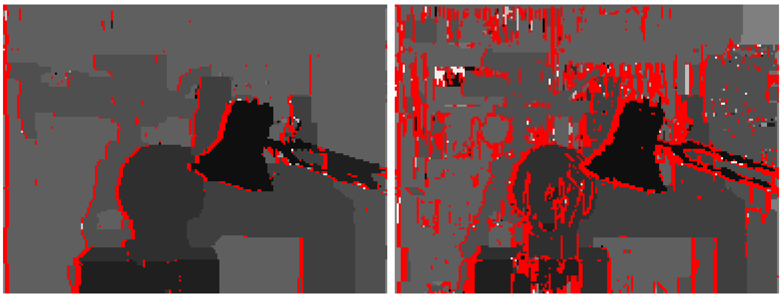


-4 to 4 -8 to 8 -16 to 16

- More disparities -> more time
- The deficit / excess in ‘granularity’ for 4to4 vs 16to16 is what results in wildly varying gross errors. What that means is,
    - -4to4: Consider most labels are between -7 to -6 in the ground truth -> very low gross error.
    - But -16 to 16 does encompass the -7 to -6 stretch -> very high gross error as chances of discrepancy being only 1 off are high.
- They do, however, have similar total errors.

2. Varying Distance Penalties

Penalty	Runtime	% Occluded	Total Err	Gross Err
B-T(default)	1m16s	1.31	36.6	22.1
MSD	45s	12.2	43.5	20.3



Tomasi Mean-Squared

Question 2

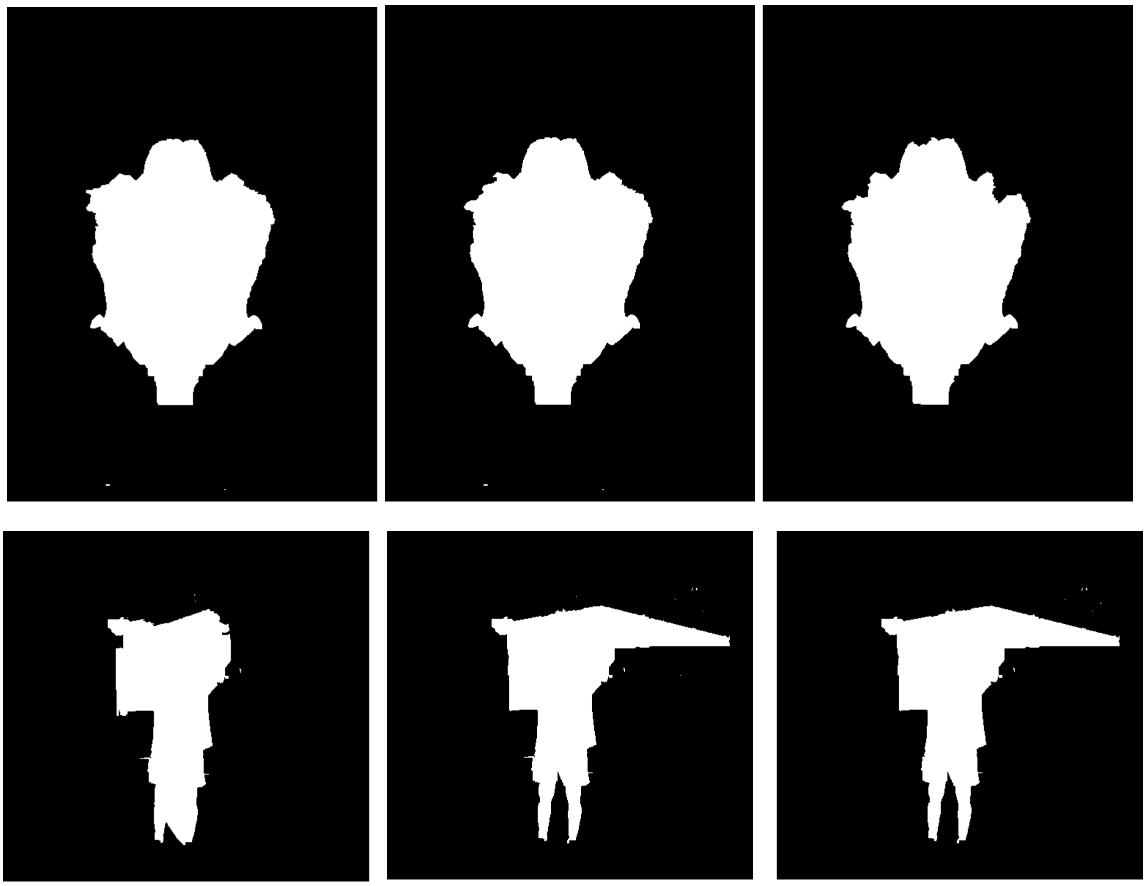
Experiment 1 - Iterations

Note: The size of bounding boxes across these different experiments was frozen for a particular image for consistency.

```
x1, y1, x2, y2 = draw_rectangle(img)
rect = (min(x1, x2), min(y1, y2), abs(x2-x1), abs(y2-y1))
# rect = (68, 153, 338, 500) # memorial
# rect=(143,27,325,452) #tennis
# rect=(117,109,356,519) #llama
```

Metrics

	Memorial			Tennis			Llama		
Iters →	2	5	10	2	5	10	2	5	10
Accu	0.990	0.990	0.990	0.894	0.932	0.932	0.842	0.841	0.842
Jaccard	0.948	0.947	0.951	0.427	0.534	0.535	0.518	0.516	0.518
Dice	0.973	0.972	0.975	0.599	0.696	0.697	0.683	0.681	0.682

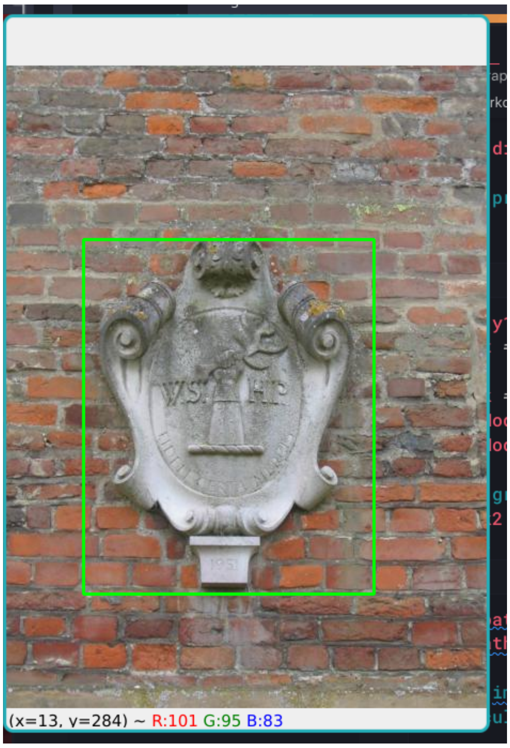


Observations

- Llama shows the least variation, with nearly identical accuracies across the board.
- In the case of tennis, a decent improvement is obtained using 5 iterations as opposed to 2, but there seems to be little to no improvement from 5 to 10. This is probably due to diminishing returns on performing further EM steps.
- ‘Finer’ detail is captured as the number of iterations is increased. For instance,
    - The gap between the person’s legs in Tennis (however, the shed in the background also gets captured at higher iters).
    - The little projections on the top left/right of the memorial are more granularly captured.

This is probably because the Gaussians used in the GMM are able to fit more specifically to the input image when they have more ‘room to breathe’ in the form of more iterations.

Experiment 2



Example of a tight bounding box

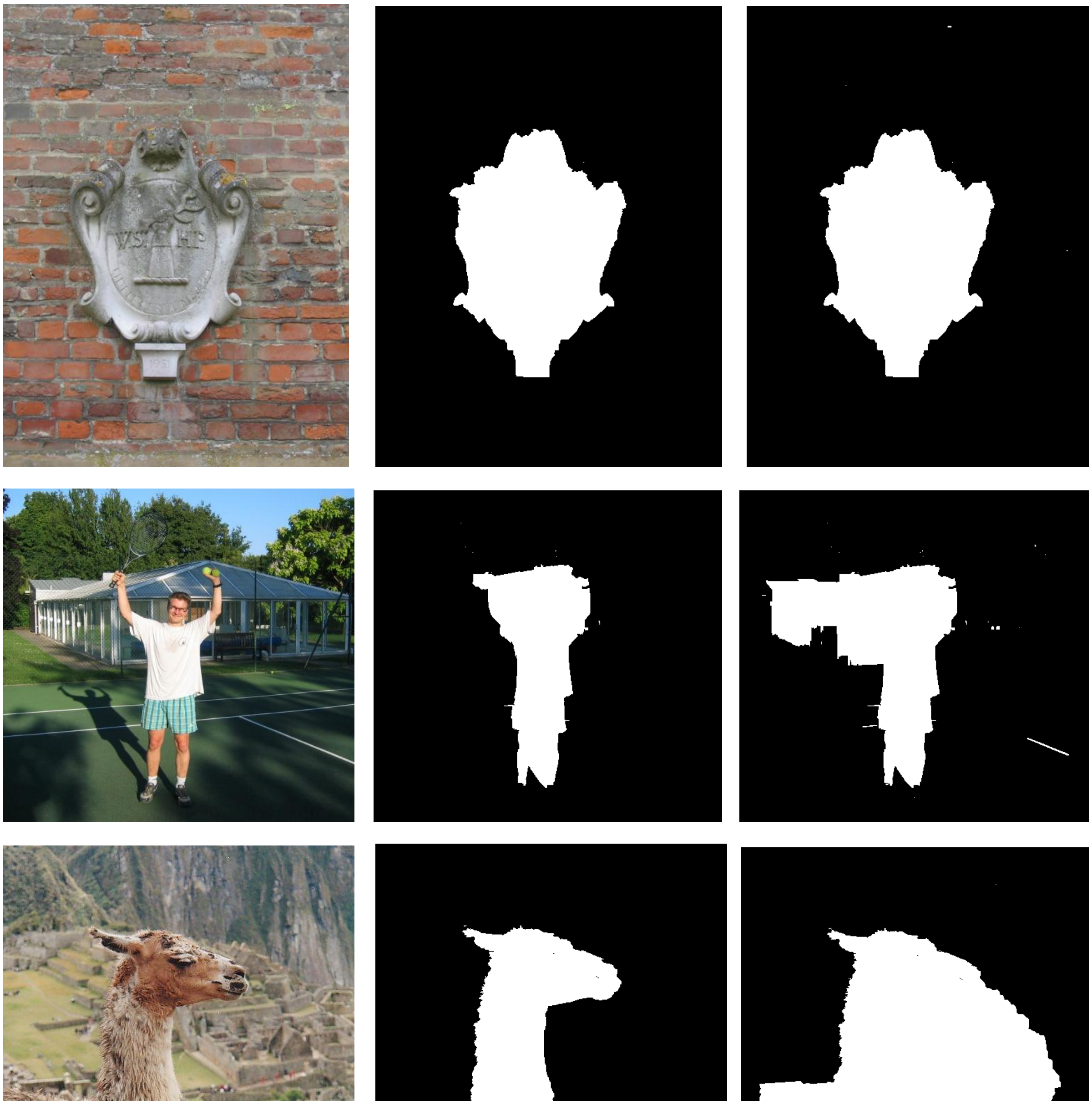
Example of a loose bounding box

Metrics

	Memorial		Tennis		Llama	
	Tight	Loose	Tight	Loose	Tight	Loose
Accuracy	0.986	0.986	0.953	0.902	0.991	0.828
Jaccard	0.925	0.925	0.624	0.442	0.951	0.498
Dice	0.961	0.961	0.768	0.613	0.979	0.664

Results

Original | Tight Bounding Box | Loose Bounding Box



Observations

- Across the board, a tighter bounding box results in better segmentation outputs.
- When the bounding box is too large, it may enclose a significant portion of the background as well as the object of interest.
- This can cause the GrabCut algorithm to include some of the background pixels in the foreground region and vice versa. A tight box provides for a better initialisation for GrabCut.

Trying Out Other Images + Experiment 3

Default config: 5 iterations, as tight a bounding box as possible.

Bounding box hardcoded between modifications to ensure objectivity between comparisons.

Banana2 - Works Well

Image	Modification	Accuracy	Jaccard	Dice
Banana2	None	0.990	0.961	0.980

Original Image

Segmentation mask👍

Banana3 - Works Well (T&C apply)

Image	Modification	Accuracy	Jaccard	Dice
Banana3	None (5 iter)	0.919	0.743	0.853
	10 iterations	0.919	0.929	0.963

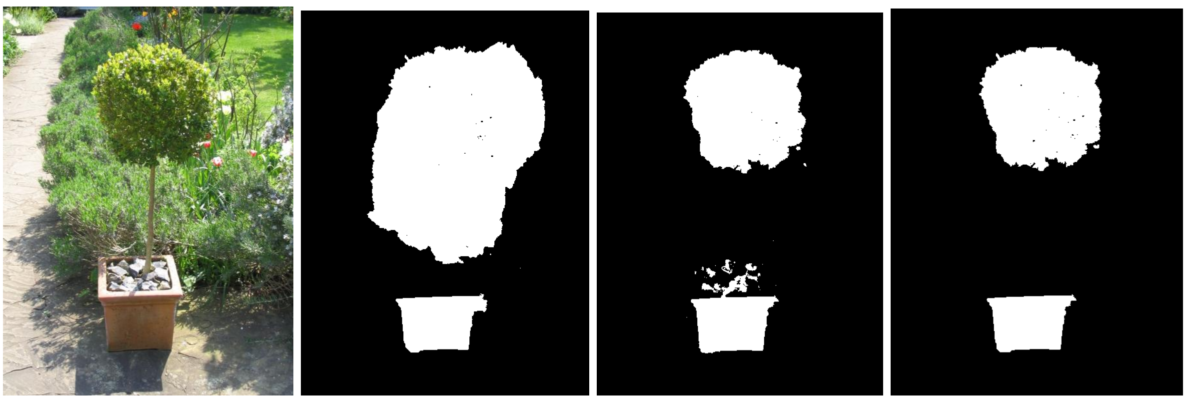
Original Image　Segmentation (5iter)　Segmentation (10iter)

- Probably due to the busy background, GrabCut seems to have some trouble fitting the Gaussians within the allotted number of steps.
- Therefore, my intuition was to give it more room to breathe, hopefully, then it would fit correctly. And it did.
- Running GrabCut for 10 iterations fixes the problem and gives a much better segmentation, as is evident from the improved metrics.

Bush - Does Not Work Well

Image	Modification	Accuracy	Jaccard	Dice
Bush	None	0.922	0.441	0.612
	HSV space	0.966	0.809	0.894
	LAB space	0.960	0.769	0.869

Original Image　Segmentation (RGB)　Segmentation (HSV)　Segmentation (LAB)



- Here, it was apparent to me that the reason for a chunk of the background didn't simply have to do with 'not enough available steps. Instead, it was probably because of the color similarity between the actual FG and BG.
- Therefore, I decided to try out different color spaces here.

Alternate Color Spaces

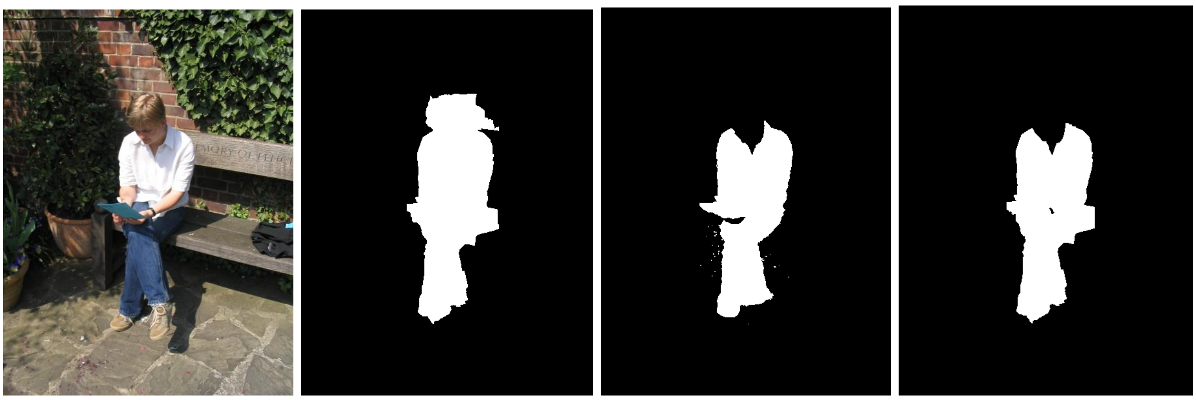
- The **CIELAB color space** separates color information into three channels: L (lightness), a (green-red), and b (blue-yellow), allowing the algorithm to differentiate between objects with similar color but different brightness or contrast.
- The **HSV color space** separates color information into hue, saturation, and value components, which can be useful in cases where the color information is more important than the brightness information.

For the bush image, both the alternate color spaces are able to capture foreground-background separation very well. HSV results in some noise near the soil, while LAB is very clean-cut. But this ‘noise’ is nothing but the pebbles in the pot, resulting in a marginally higher accuracy for HSV over LAB. However, both of them are significantly better than RGB.

Person6 - Does Not Work Well (T&C Apply)

Image	Modification	Accuracy	Jaccard	Dice
Person6	None	0.974	0.792	0.884
	HSV space	0.969	0.731	0.845
	LAB space	0.966	0.714	0.833

Original Image Segmentation (RGB) Segmentation (HSV) Segmentation (LAB)



Same problem here. RGB results in poor FG-BG separation. Using alt color spaces gives a visually better result (very clean-cut), but the accuracies are lower owing to the head being segmented out, which is rather unfortunate.