Assignment 3

a) Mask generation:

* For the image of the bird, we observe that there is a stark contrast between the foreground and the background of the image. This is maintained in the greyscale version of the image.
* We observe there to be an edge which demarcates the bird object. Hence, we use the canny edge detection algorithm on the grayscale version of the image. This detects edges in the waves of the water as well. Hence, we use only an approximation to Canny, which detects stronger edges at lower computational cost. This, coupled with thresholding with appropriate parameters [0.2 0.6], (low, high) detected the outline of the bird appropriately. Gaps were filled in by the imclose operation, bridging them by morphologically closing them using discs etc. (like a connect the dots). And then imfill, to 'colour in' the mask.
* It is a similar strategy for the mask for the flower image, though we use Canny edge detection, not an approximation.

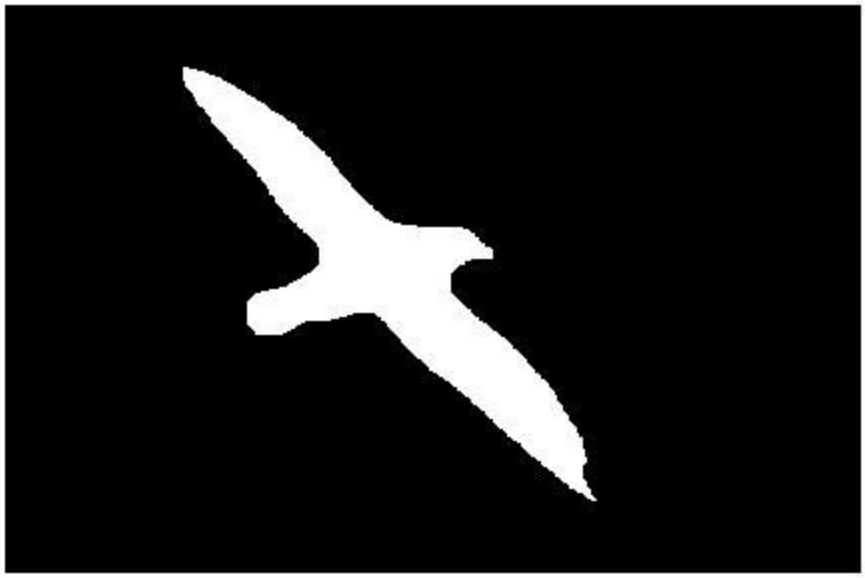
And, due to the colour of the flower, K-means clustering smooths out the background

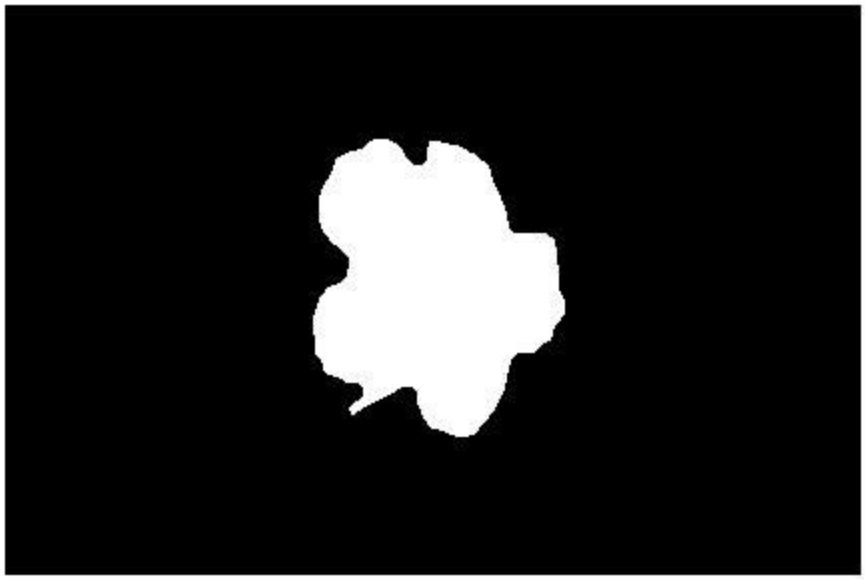
and leaves clear edges between the flower and the background. It also helps by bluring the other sharp, in-focus parts of the foreground, leaving the flower as a distinct entity.

* As it ~~is~~ also highlights the other flower/partial flower, we filter that out. Hence, we have a partially automatic algorithm here, which successfully masks the flower.

Original Images:

The masks generated are as follows:

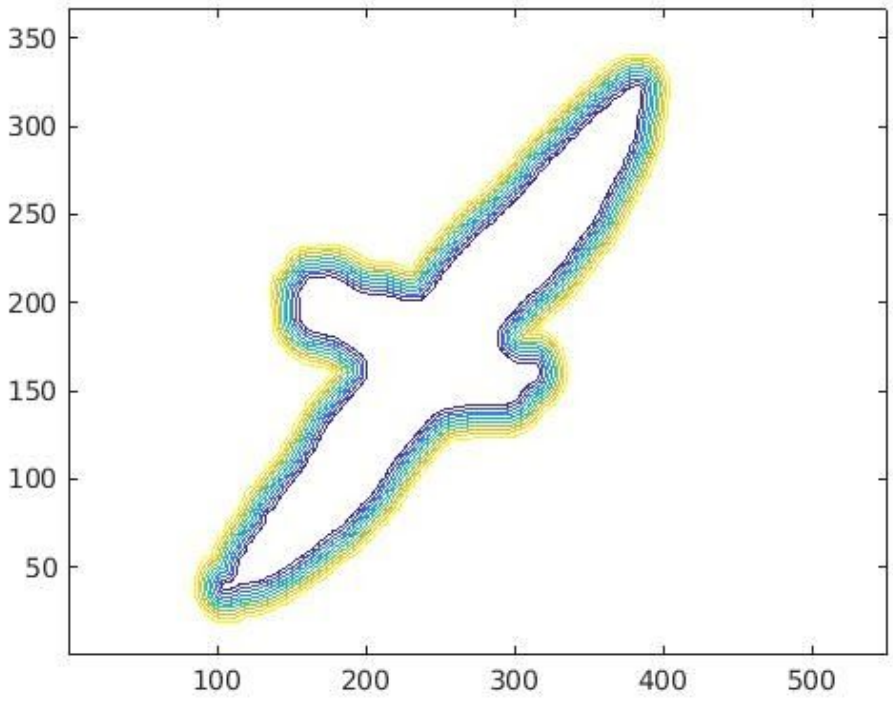




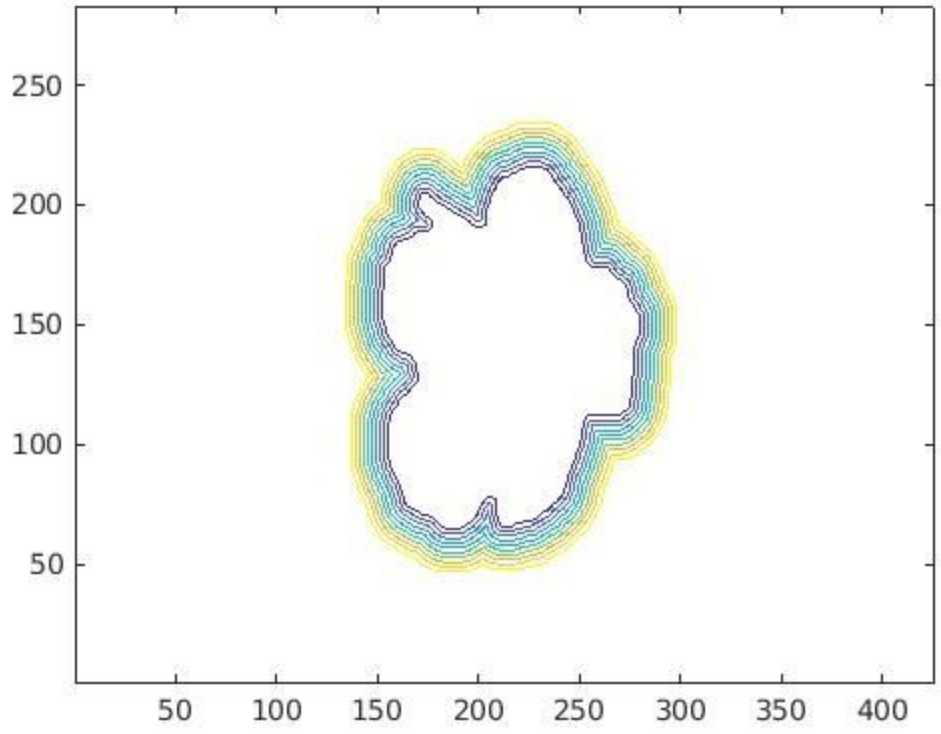
Using the above masks, we partition/segment the two images as follows:
For the bird:

For the flower:

Now, we look at the contour plot for the value of r, as the distance from the foreground increases.
For the bird:



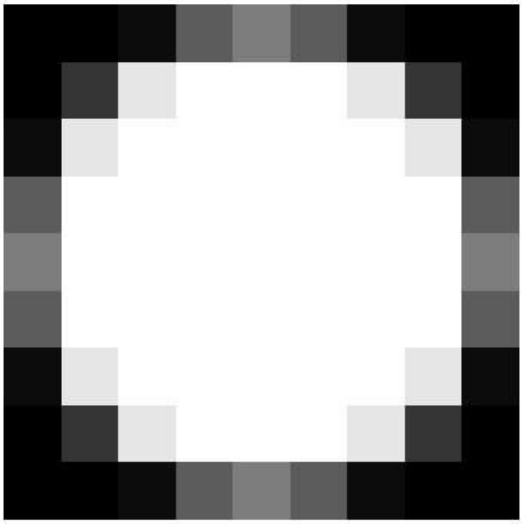
For the flower:



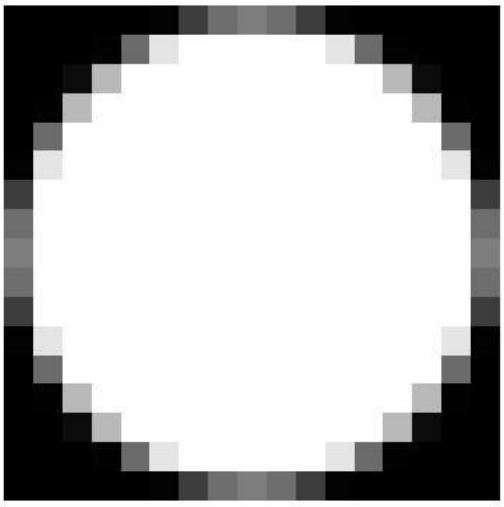
The Blurring Kernels:

For Flower: (alpha = 20)

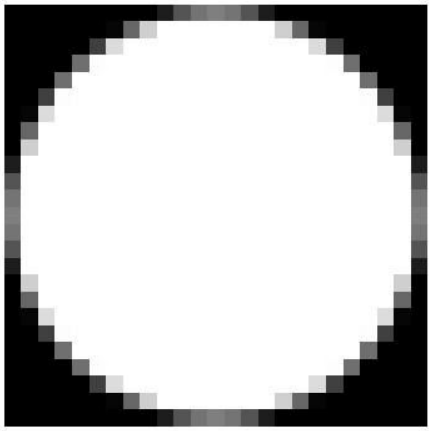
0.2alpha:



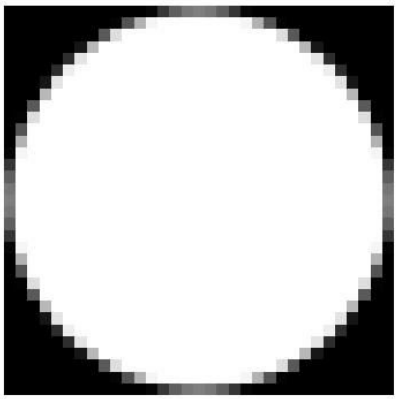
0.4alpha:



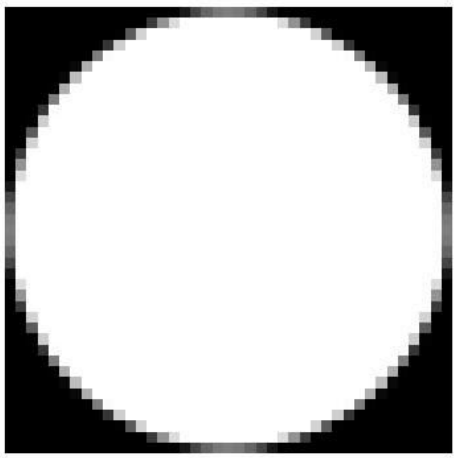
0.6alpha:



0.8alpha:



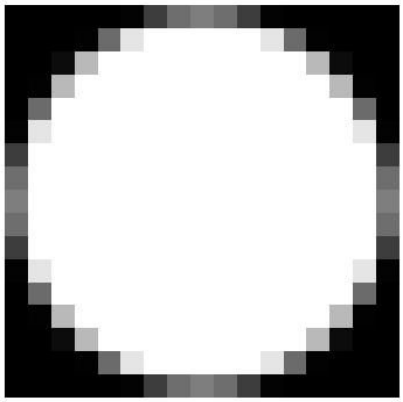
alpha:



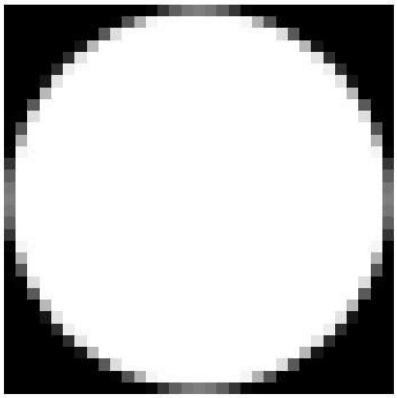
For Bird (alpha = 40):

(For implementation though, both alphas have been set to 20, quick computation. This is just to show the blurring kernels)

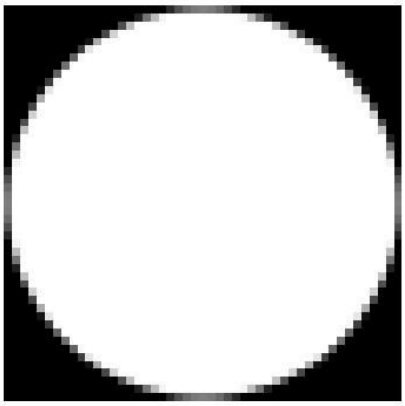
0.2alpha:



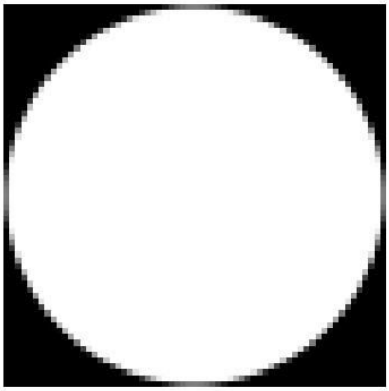
0.4alpha:



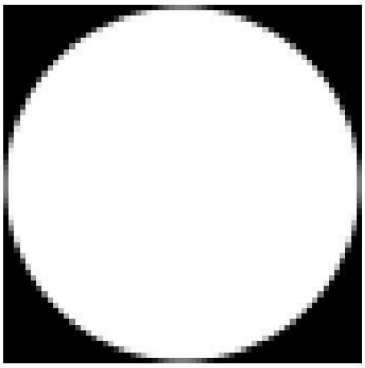
0.6alpha:



0.8alpha:



alpha:



Blurred Images: