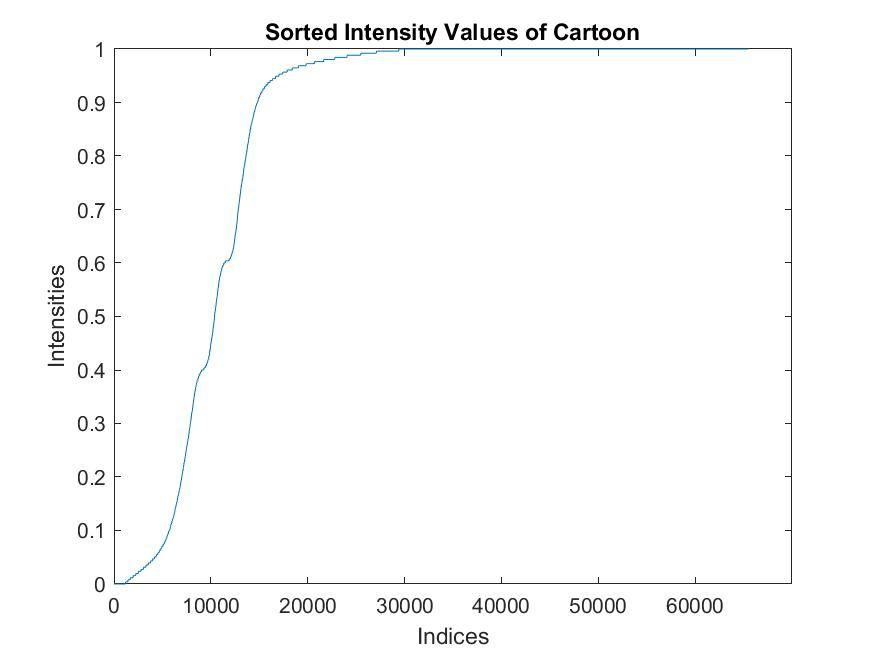
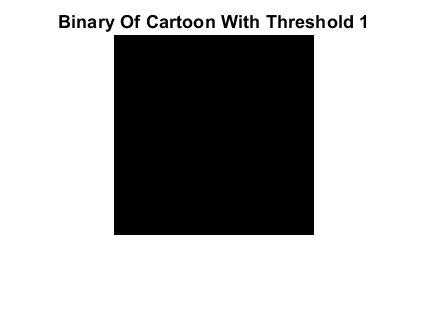
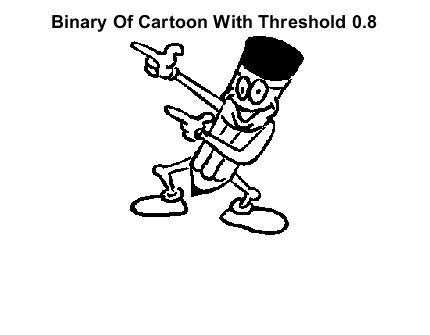
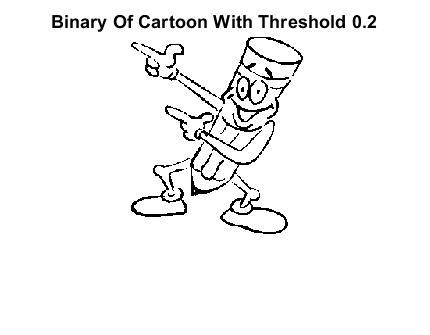
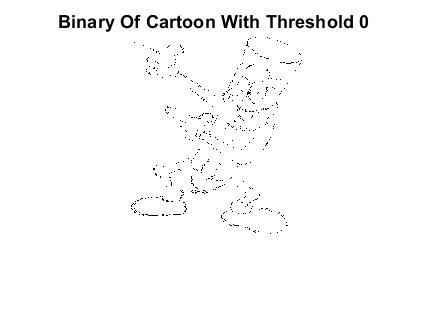
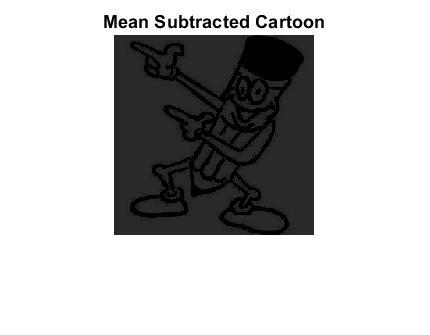
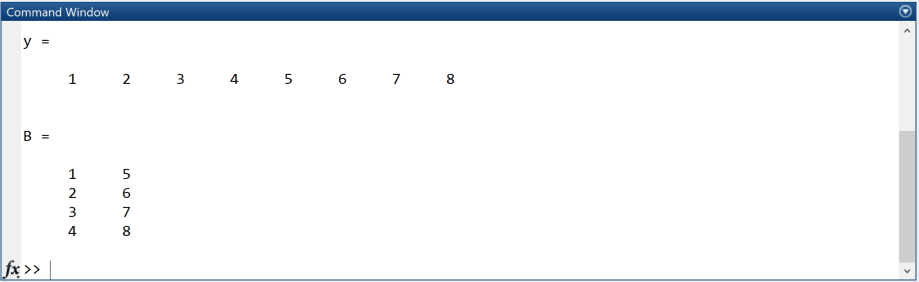
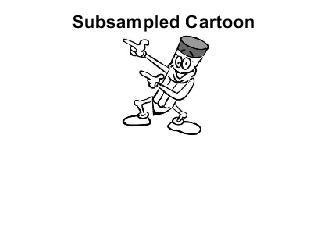
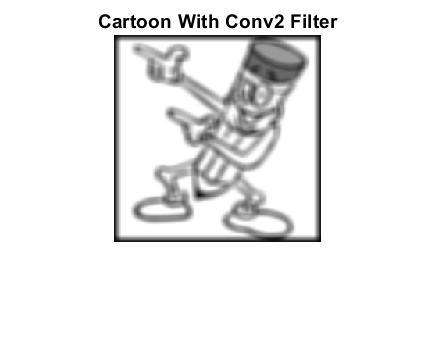
Beomjun Aaron Bae, Sakshi Agarwal

CS211A: Visual Computing

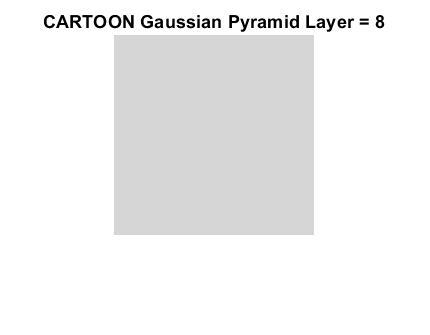
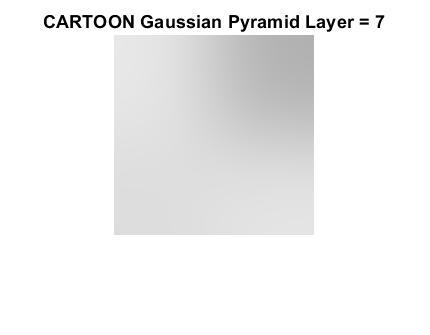
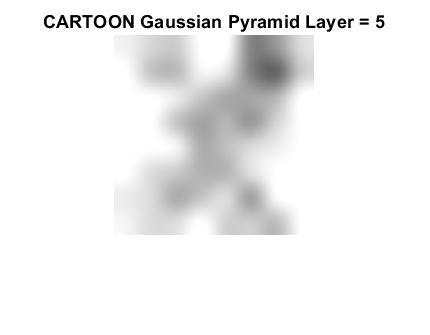
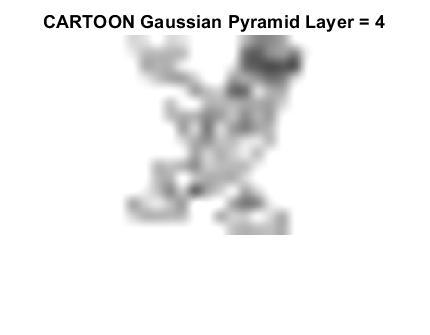
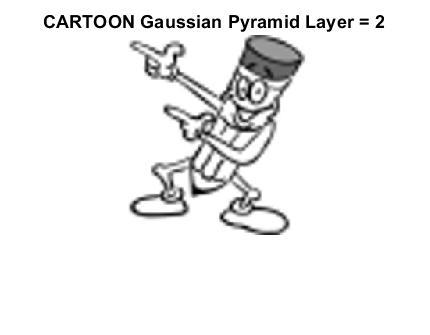
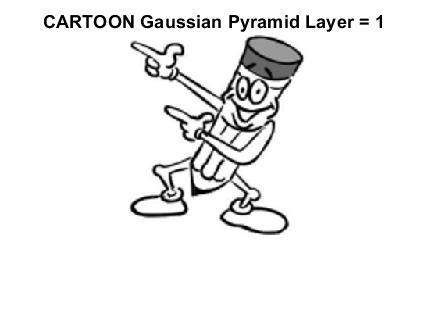
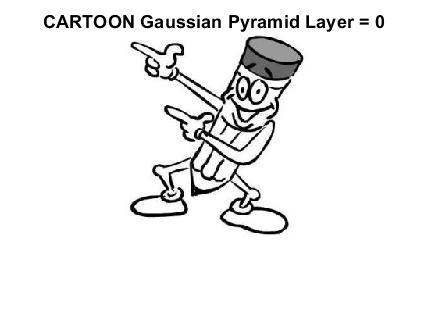
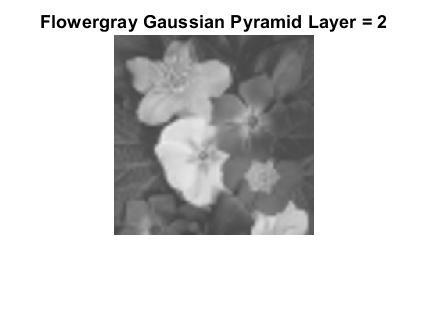
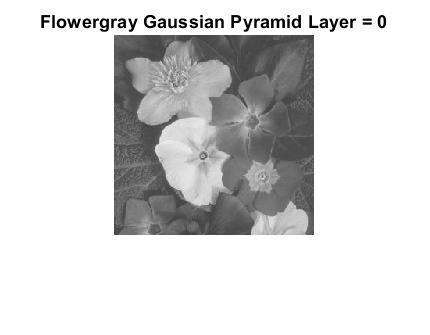
Professor Aditi Majumder

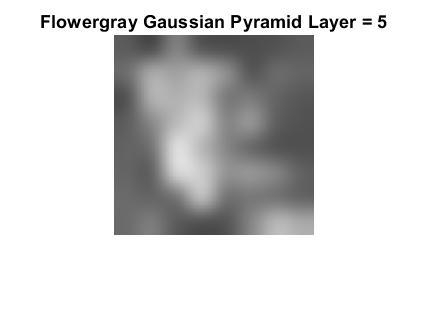
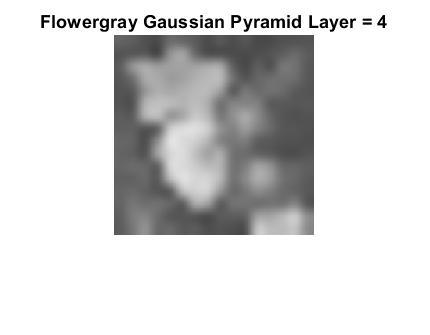
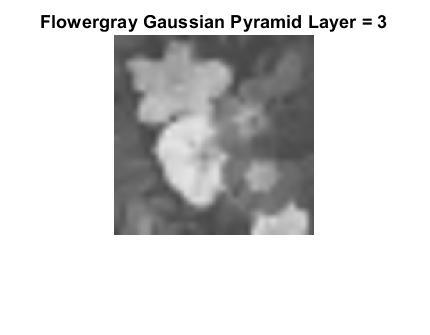
HW1 – Image Processing

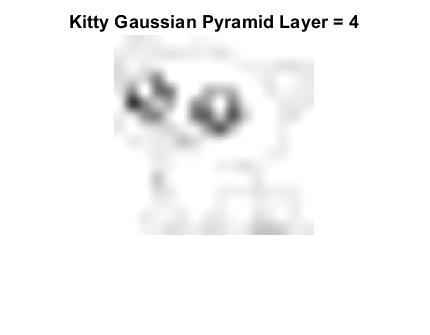
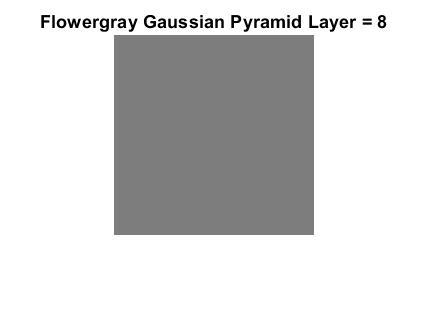
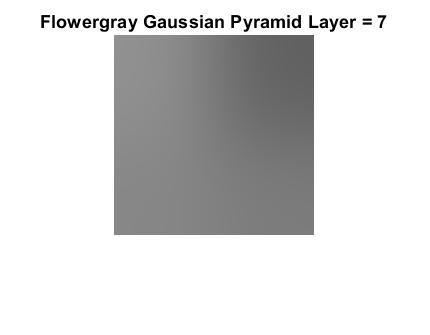
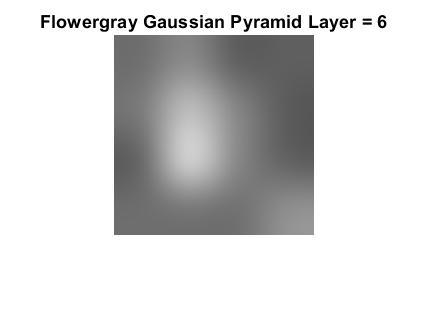
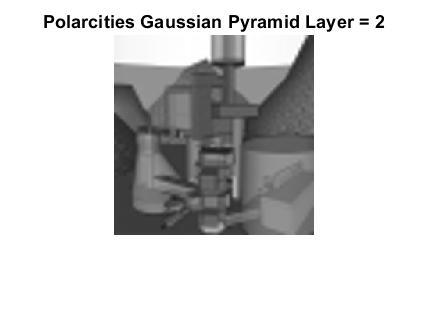
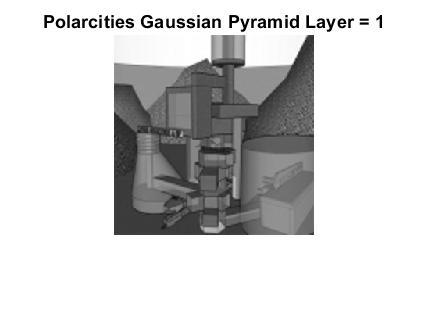
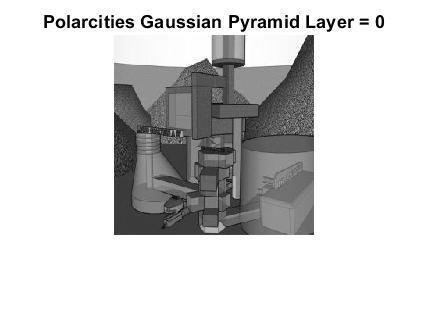
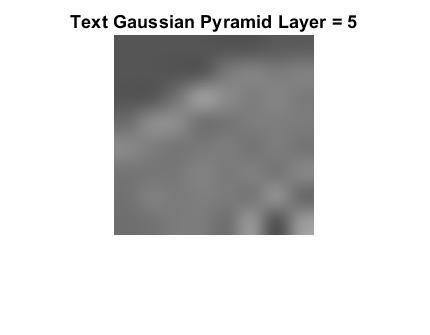
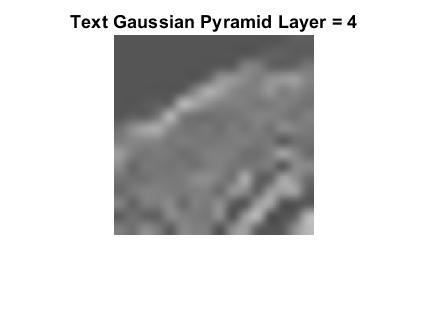
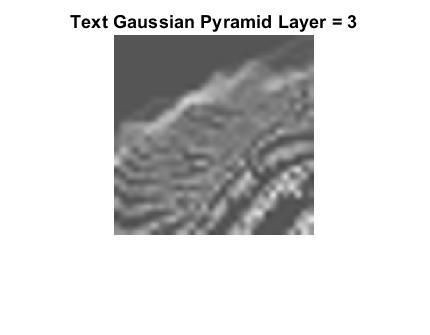
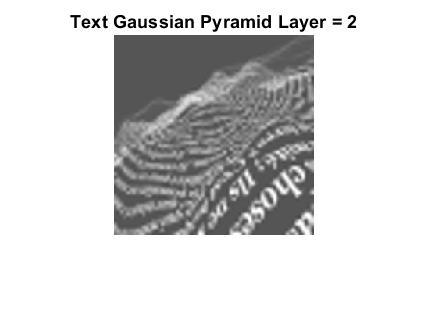
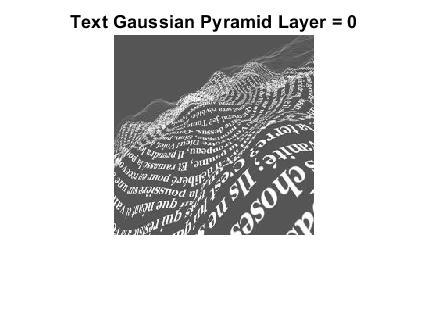
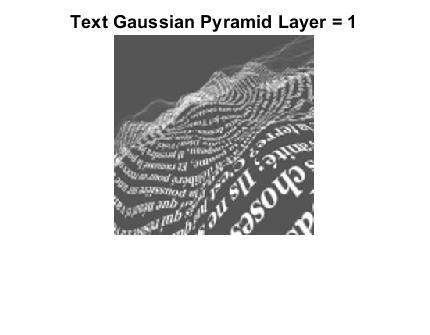
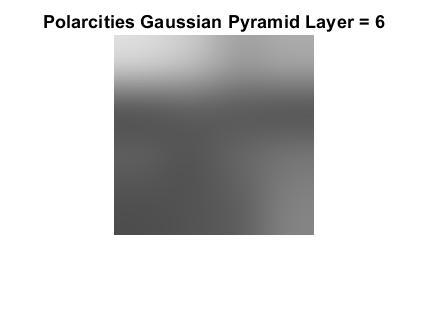
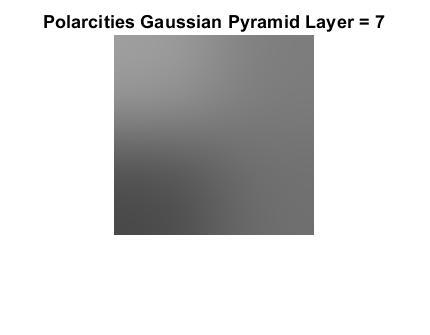
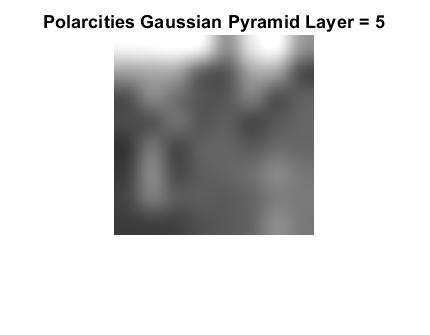
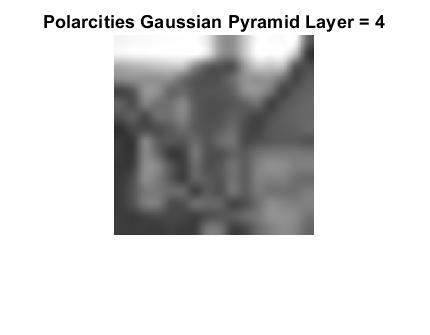
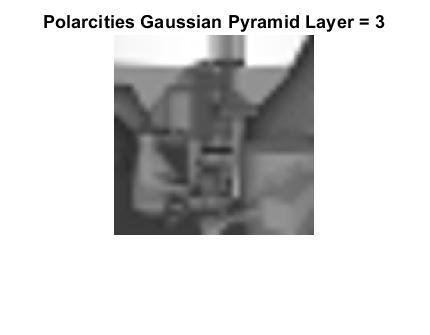
**Part 0: Getting Started**

* 1. Sort all the intensities in A, put the result in a single 10000-dimensional vector x, and plot the values in x.
  2. Display a figure showing a histogram of A’s intensities with 32 bins. 
  3. Create and display a new binary image with the same size as A, which is white wherever the intensity in A is greater than a threshold t, and black everywhere else. 
  4. Generate a new image (matrix), which has the same size as A, but with A’s mean intensity value subtracted from each pixel. Set any negative values to 0. 
  5. Let y be the vector: y = [1: 8]. Use the reshape command to form a new matrix s whose first column is [1, 2, 3, 4]’, and whose second column is [5, 6, 7, 8]’.
  6. Create a vector [1, 3, 5 …, 99]. Extract the corresponding pixel from the image in its two dimensions, i.e., subsample the original image to its half size. 
  7. Use *fspecial* to create a Gaussian Filter and then apply the *imfilter* function to the image with the created Gaussian Filter, by doing so you should see a blurred image. Change three combinations of parameters of the Gaussian Filter and compare the results. 
  8. Apply the *conv2* instead of *imfilter* function to the same process (for one Gaussian Filter), do you see any changes? Why?

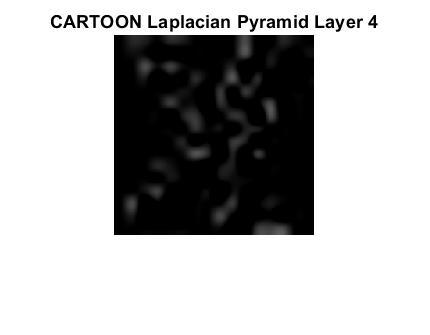
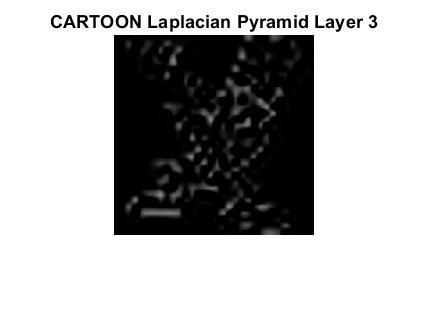
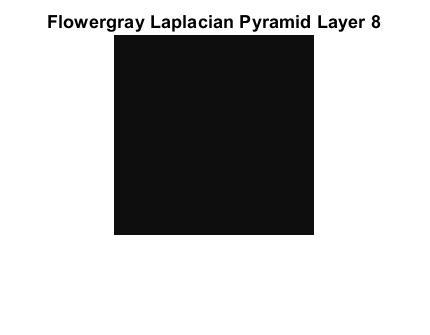
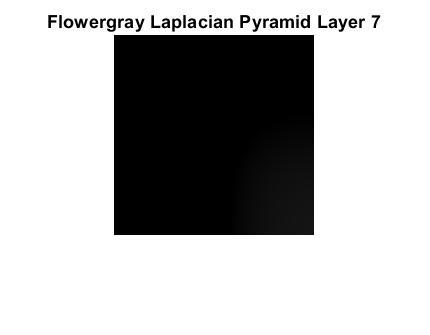
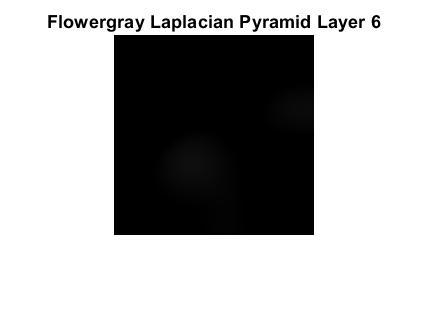
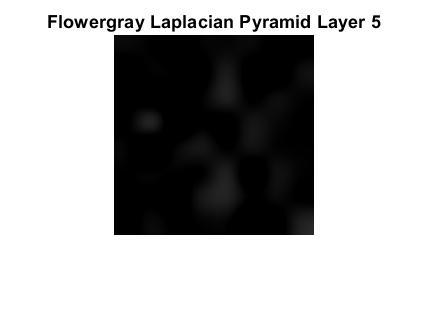
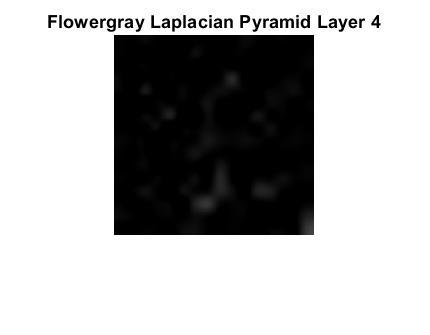
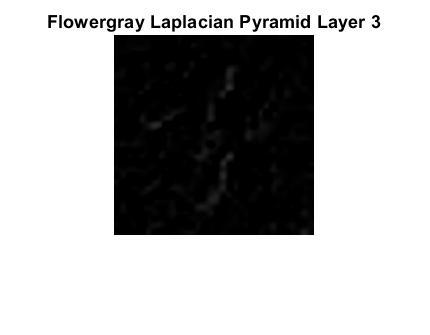
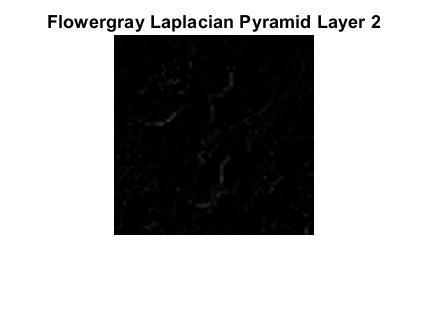
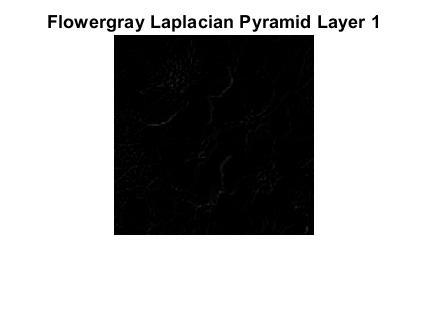
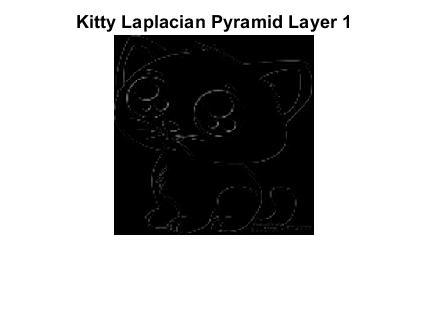
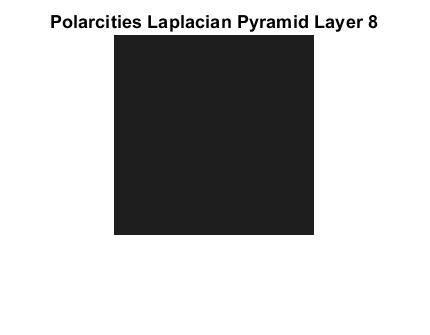
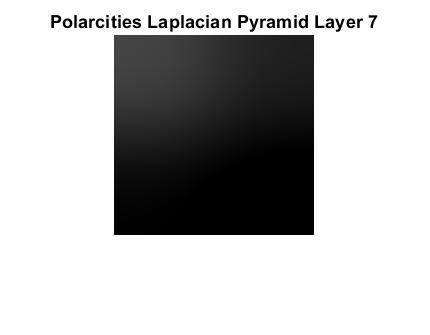
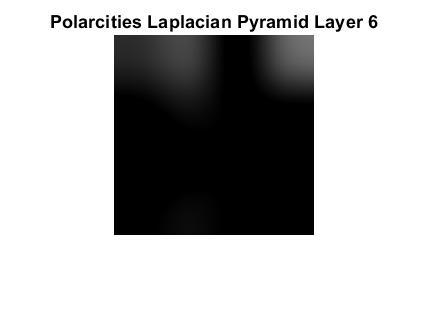
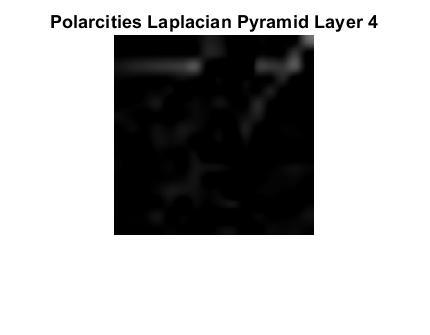
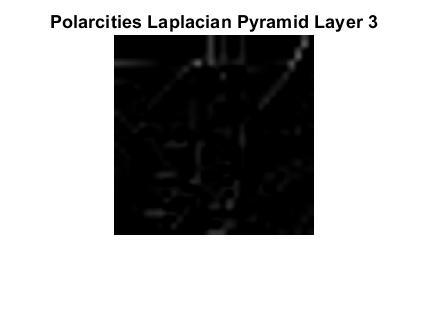
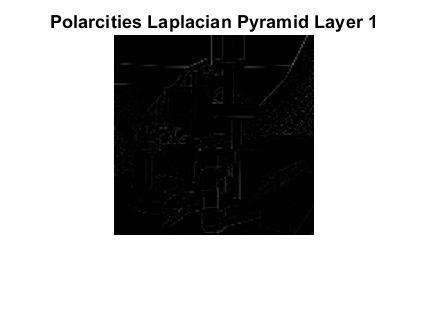
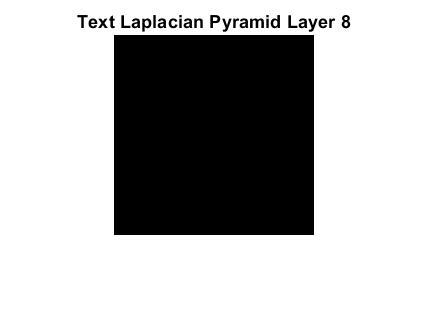
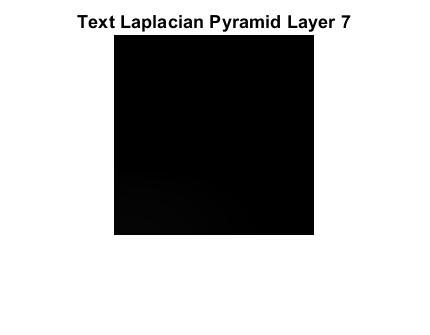
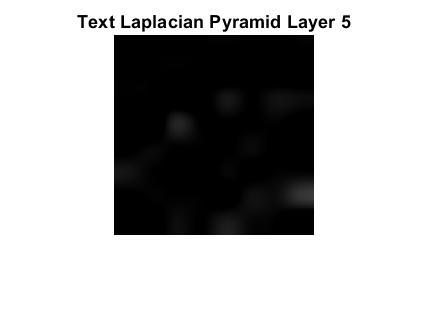
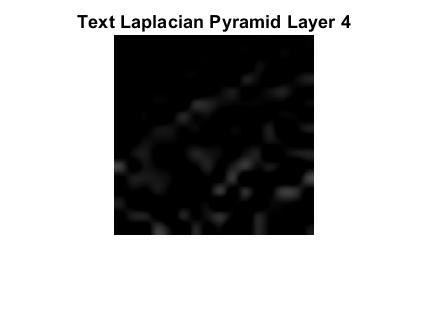
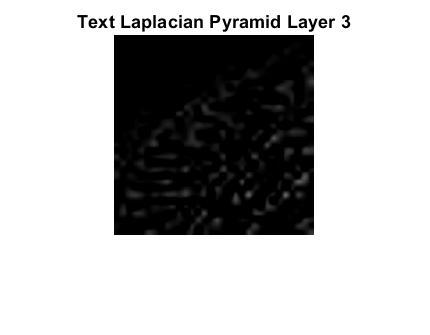
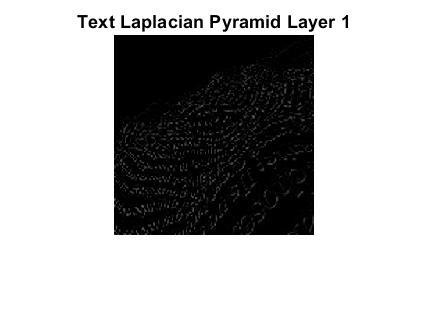
**Part 1: Gaussian Pyramid**

* Create the Gaussian pyramid for all images in the gallery and put the images in the PDF file. Please consider that all the images in different levels should have same size.
* Flowergray 



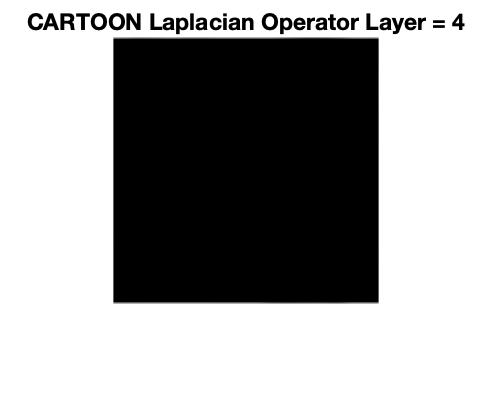
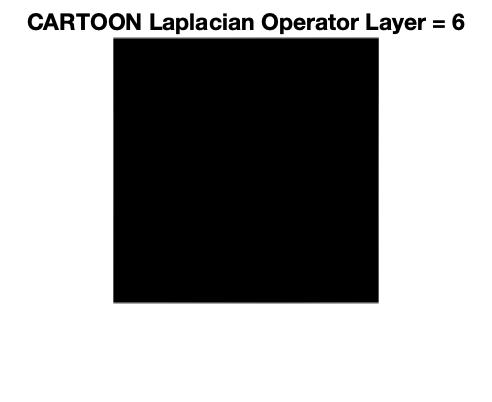
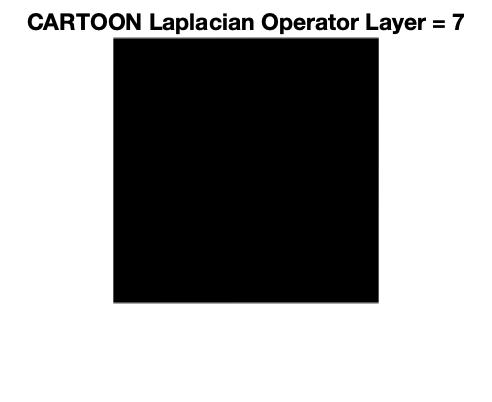
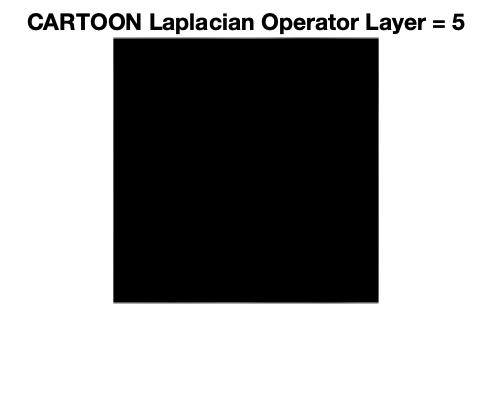
* Kitty
* Plarcities
* Text

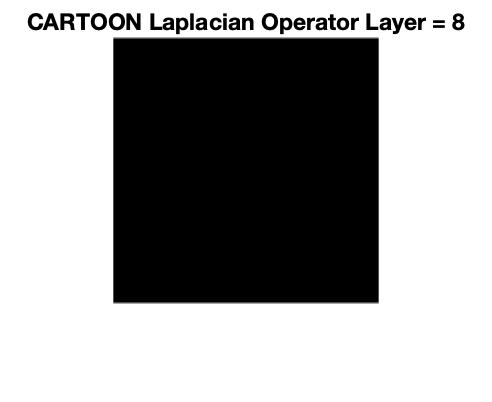
**Part 2: Laplacian Pyramid**

* Write a program to generate the Laplacian pyramid by subtracting the consecutive levels of the Gaussian pyramid. Create the Laplacian pyramid for all the images in the gallery and put the results in the PDF format.
* Cartoon
* Flower
* Kitty
* Polarcities
* Text

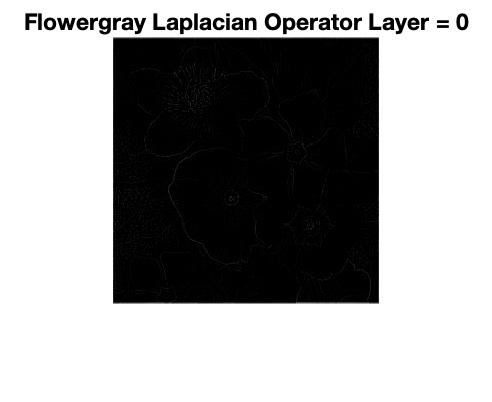
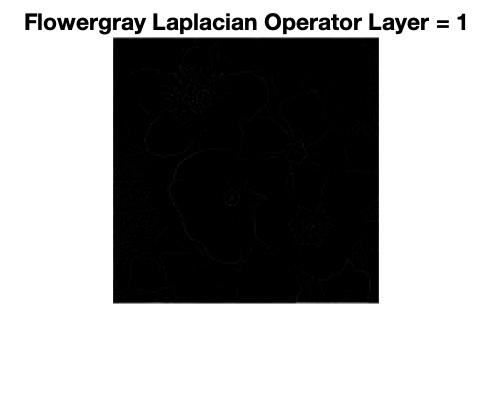
**Multi-Scale Edge Detection**

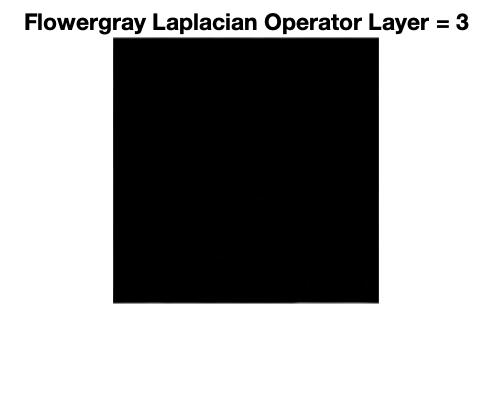
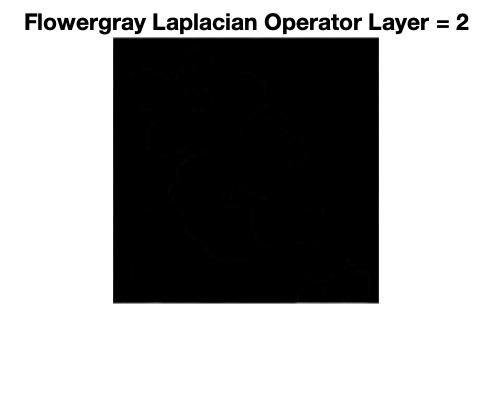
Step 1: Generate the second order derivative images at different scales (or resolution) using a Laplacian operator given below:

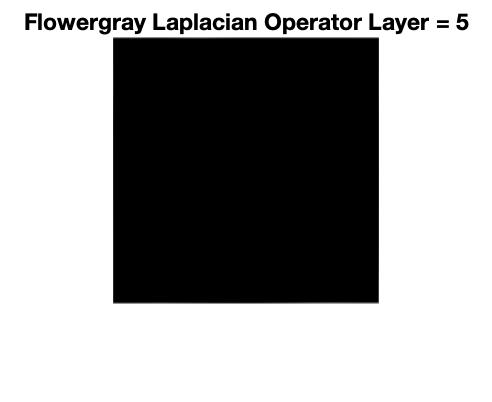
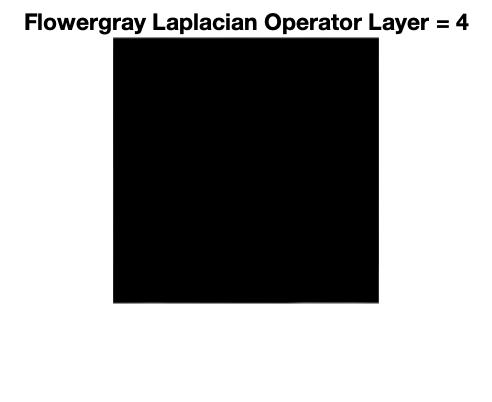
Cartoon 

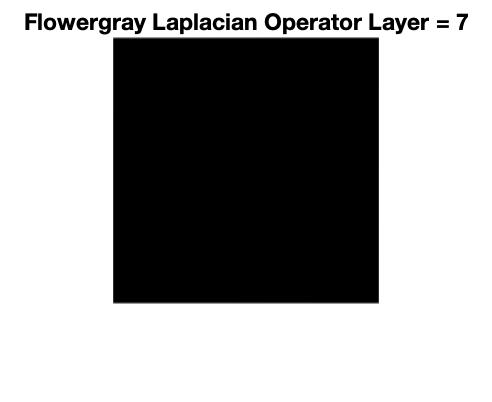
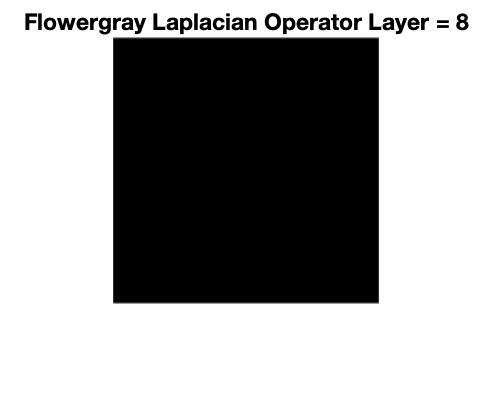


Flower









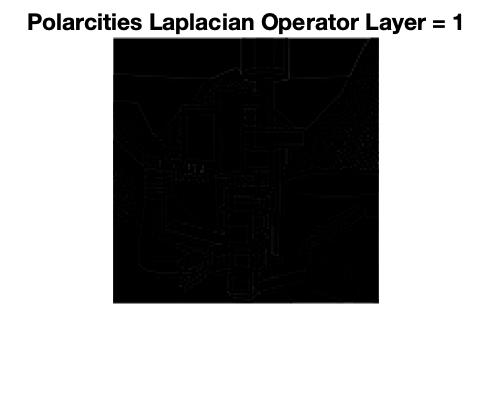
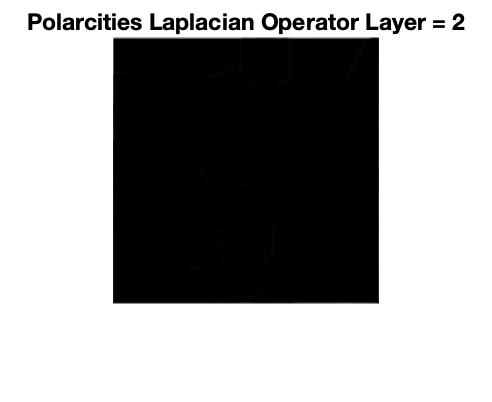
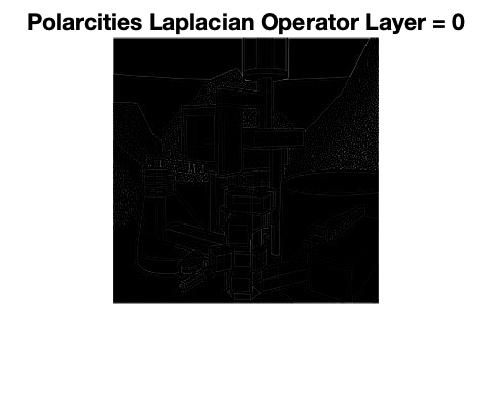
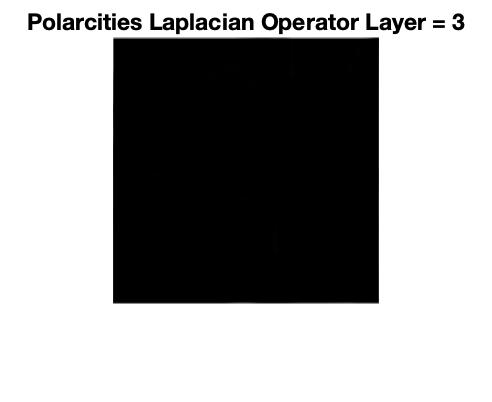
Kitty

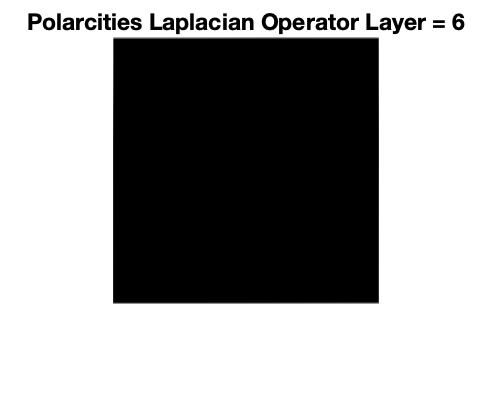
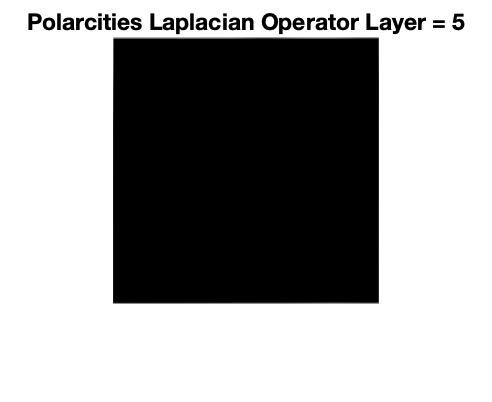
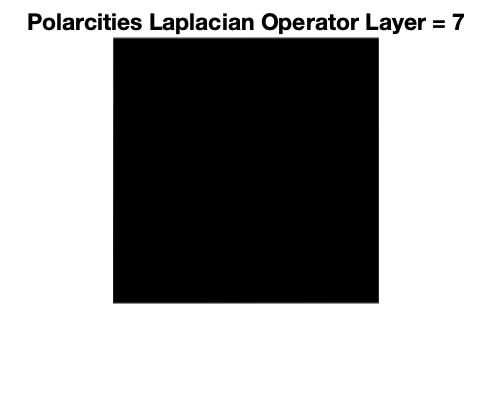
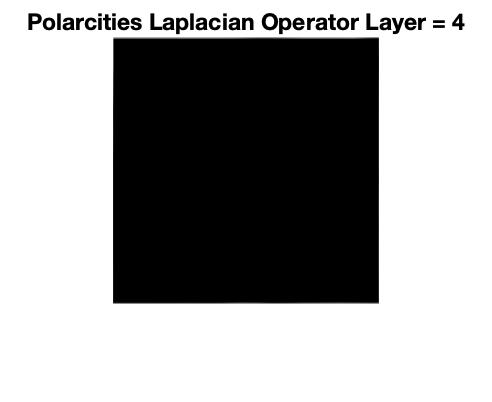




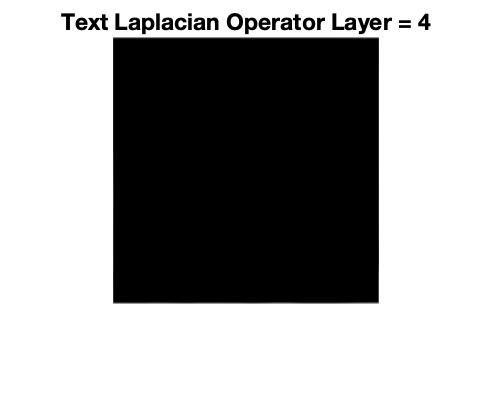
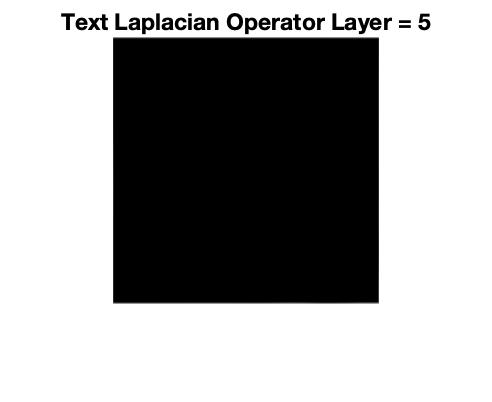


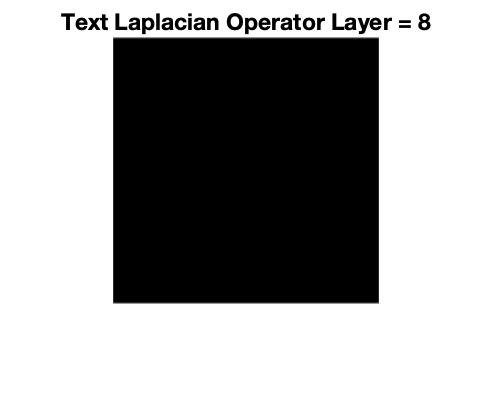
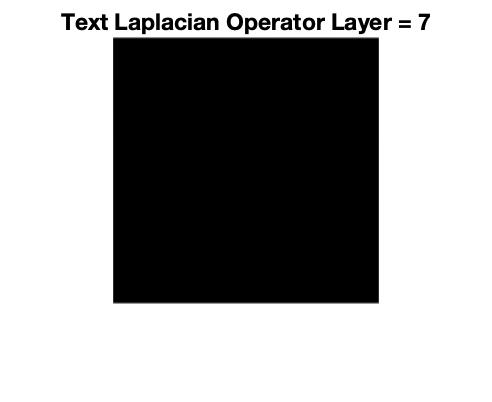
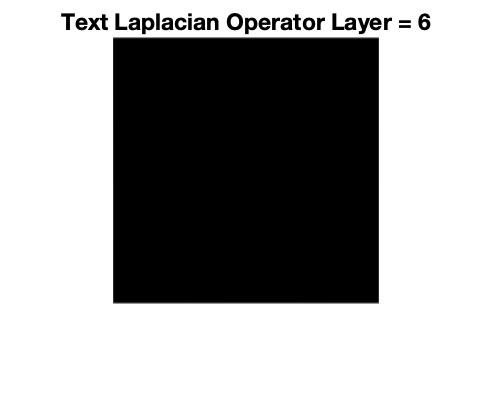


Polarcities

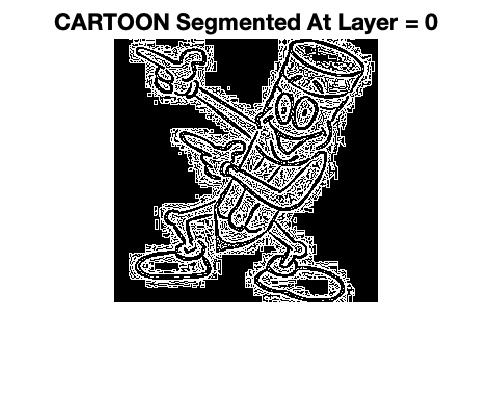
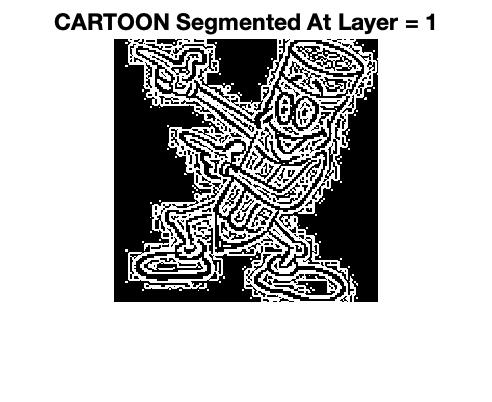


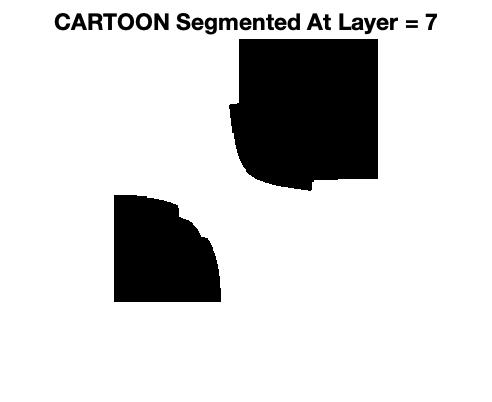
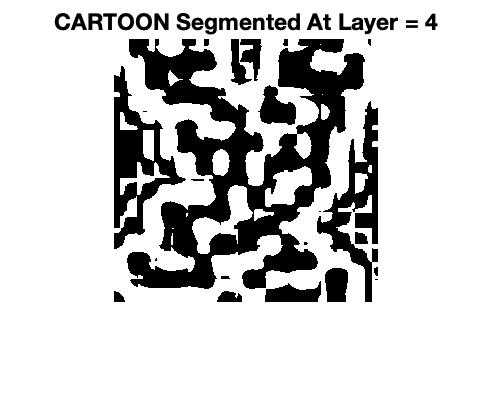
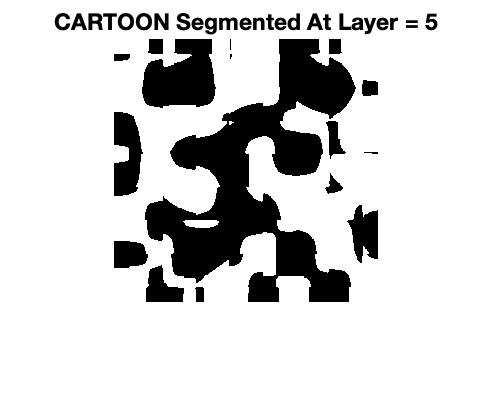
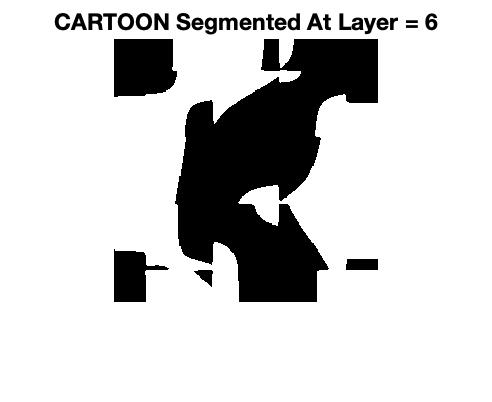
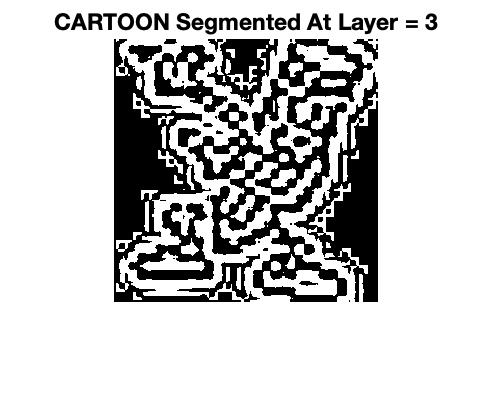
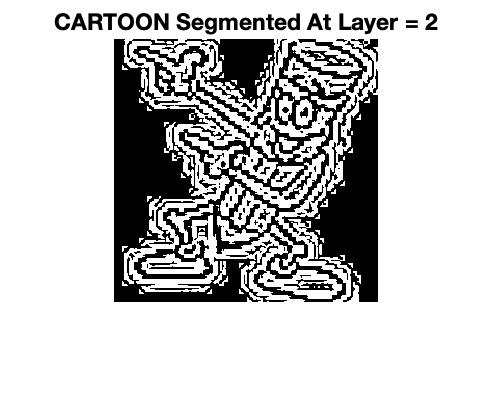
Text

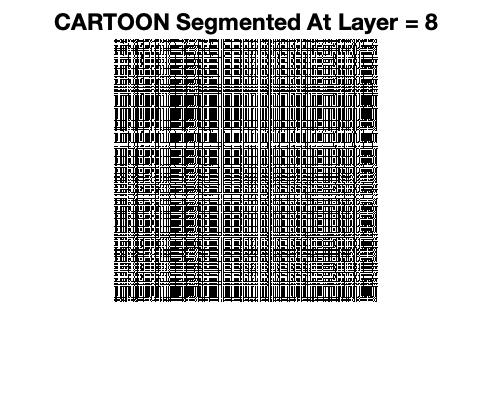


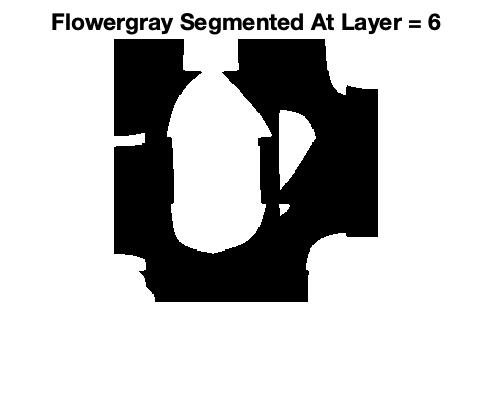
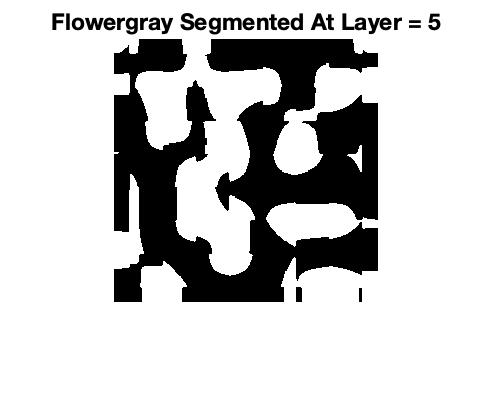
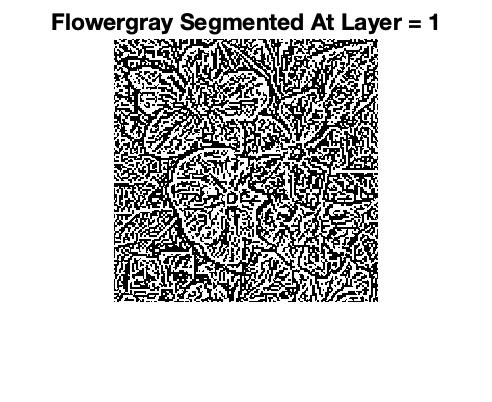
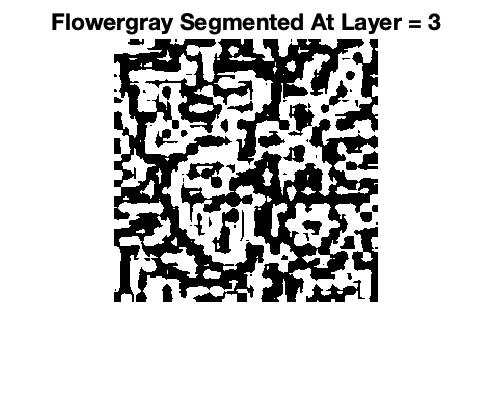
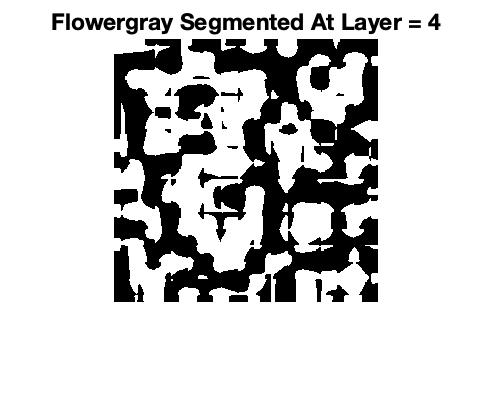
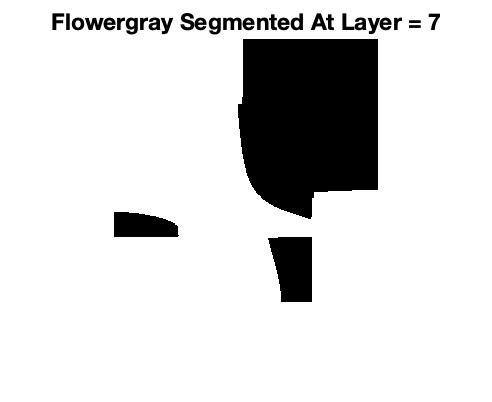
Step 2: Segment the second order derivative image by assigning value to 1 to all pixels of magnitude greater than 0 and value 0 to all pixels of magnitude less than or equal to zero. 

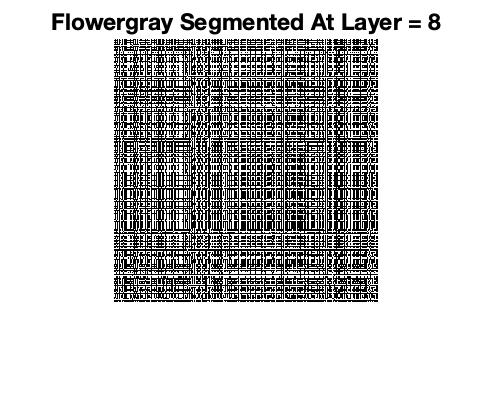
Cartoon



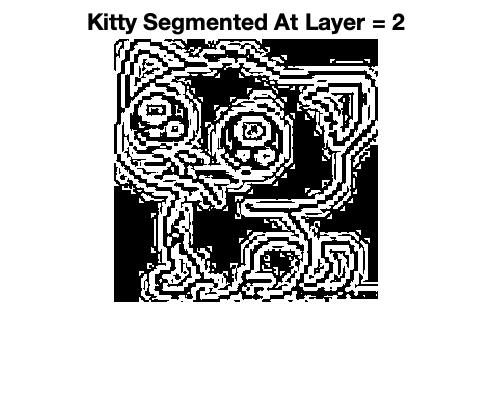
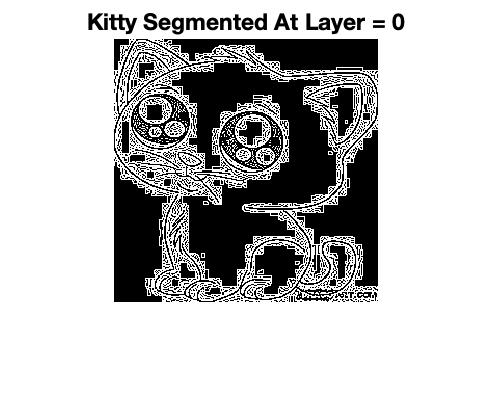


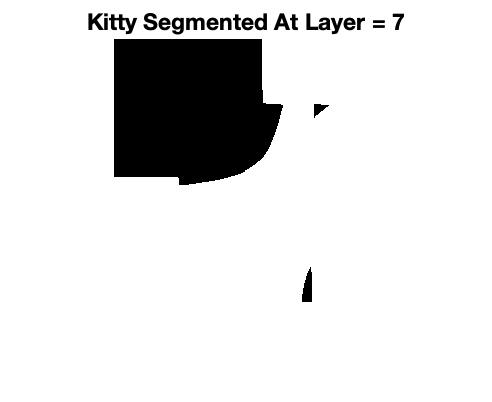
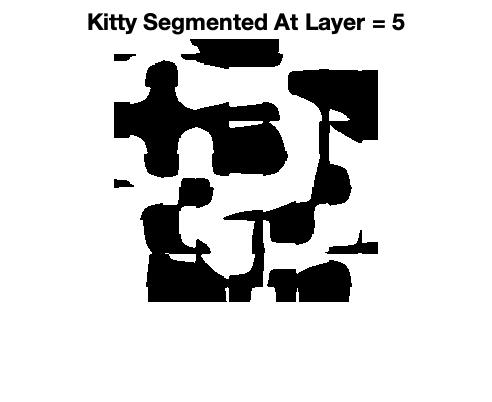
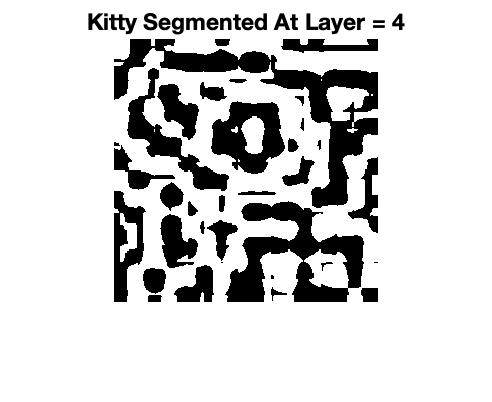
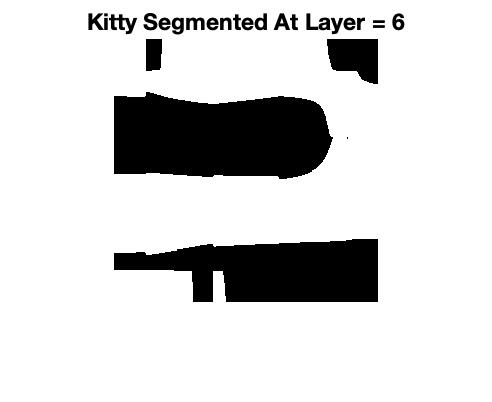
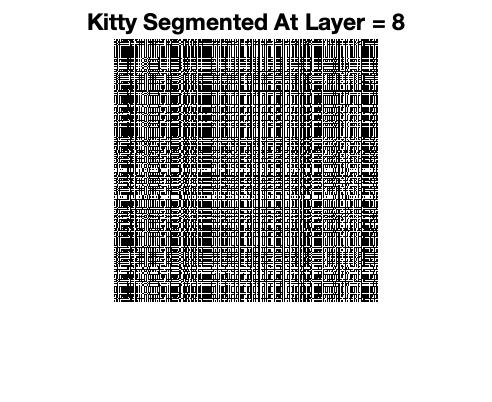


Flower

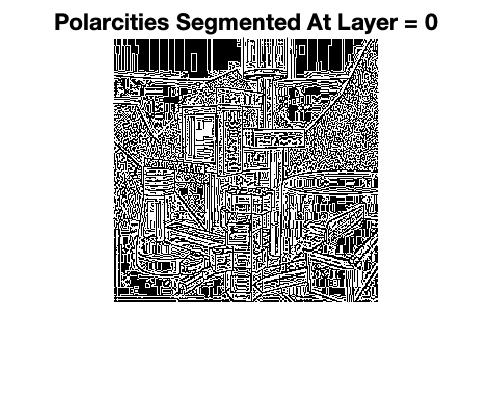


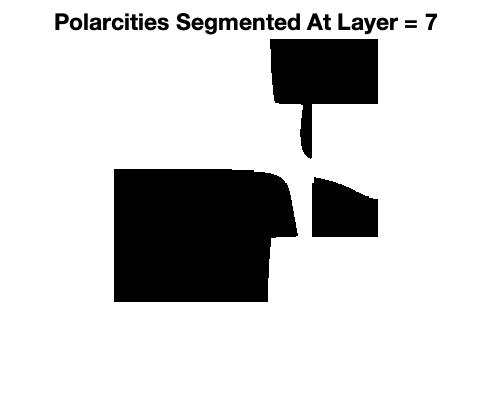
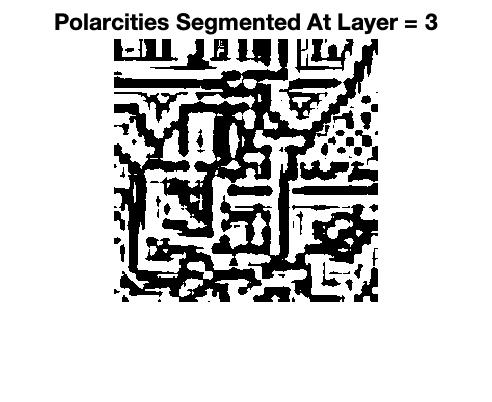
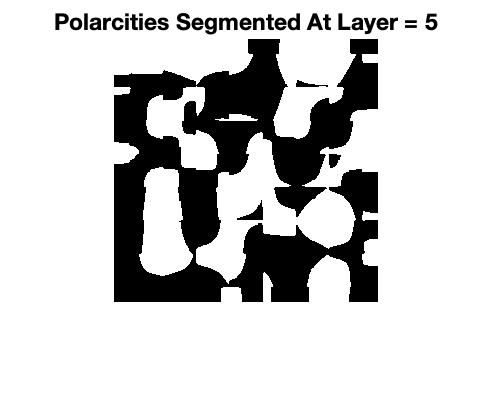
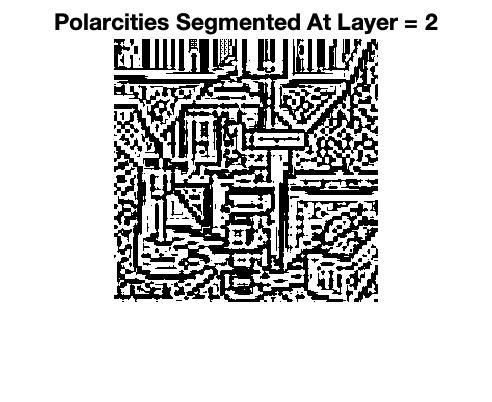
Kitty



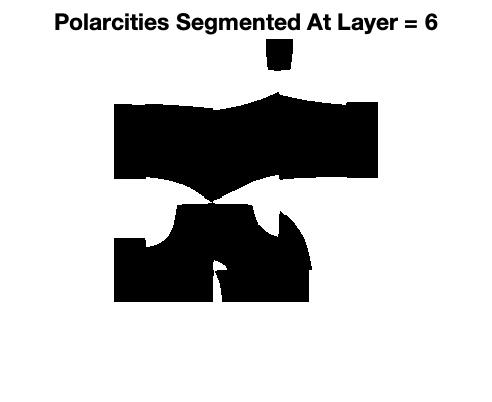


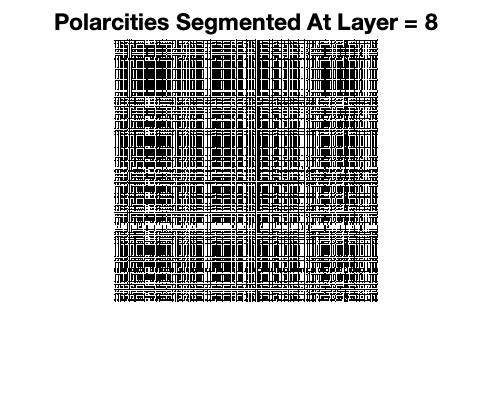
Polarcities

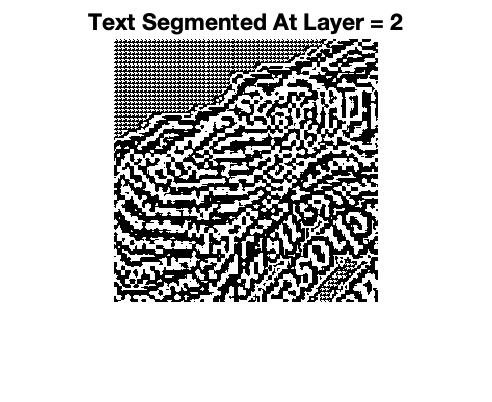
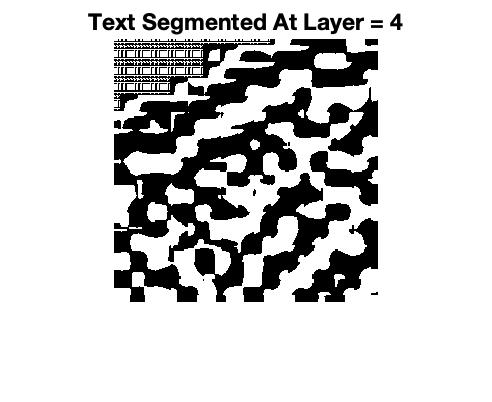
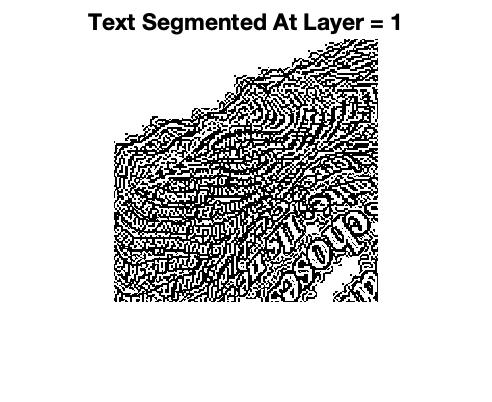
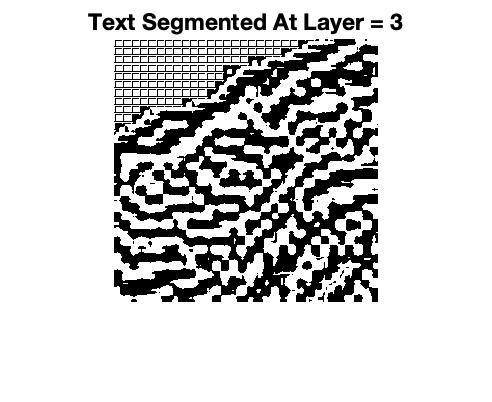
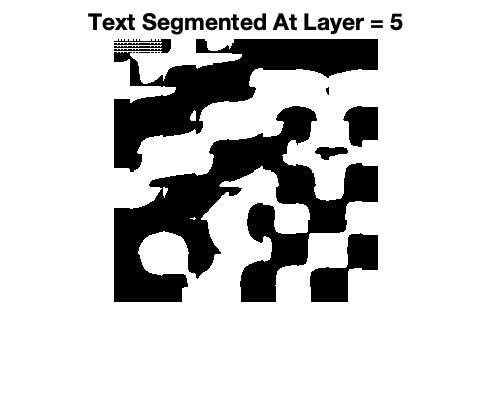
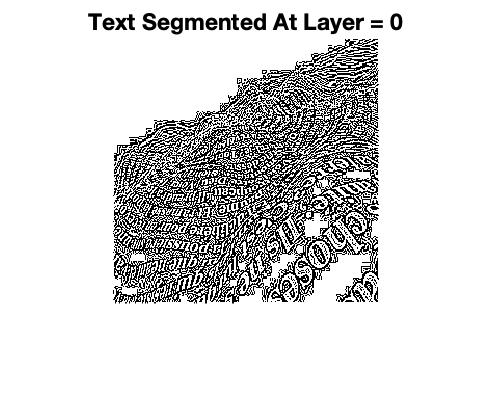


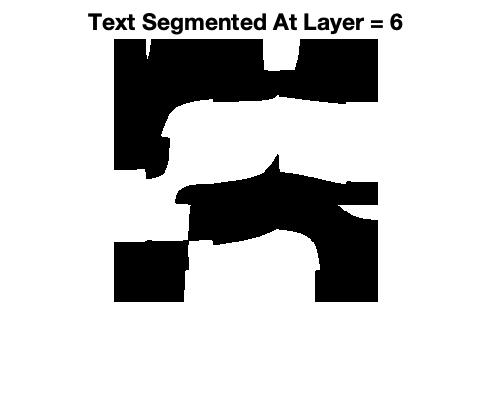
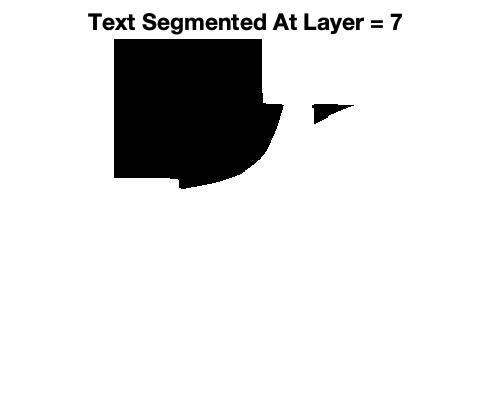
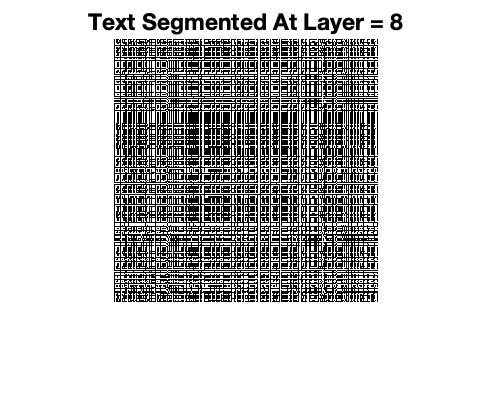


Text

Step 3: 

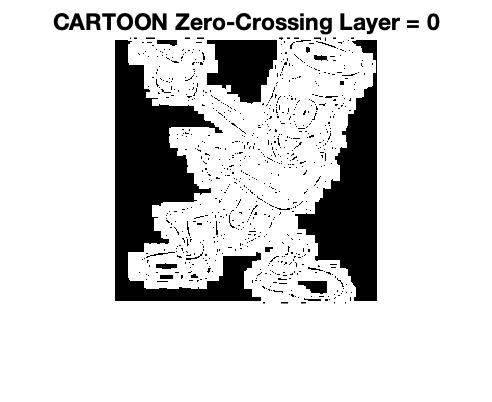
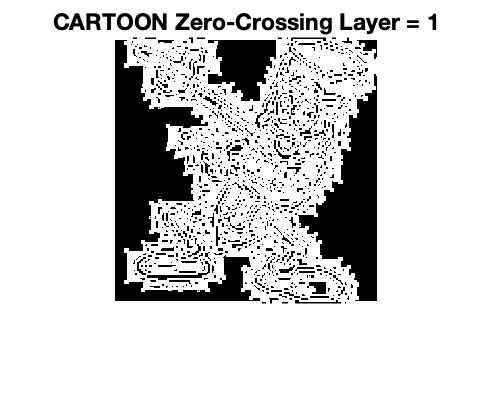
Text 

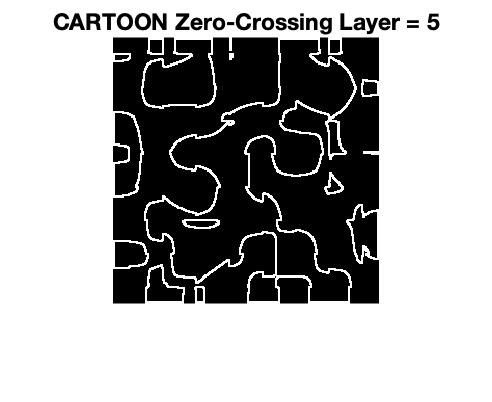
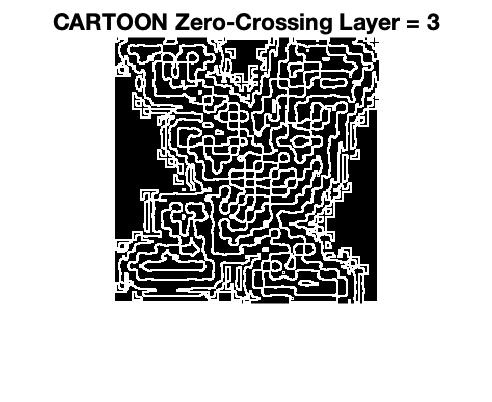
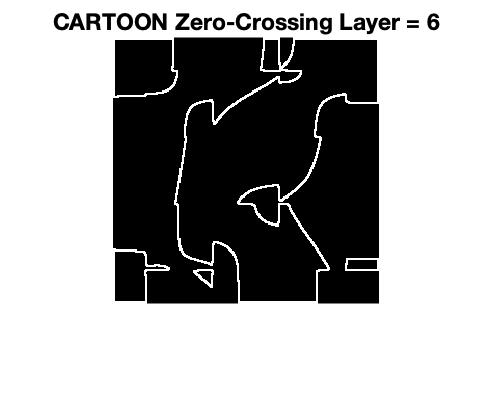
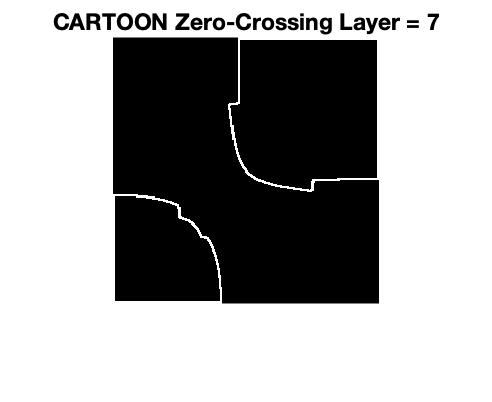
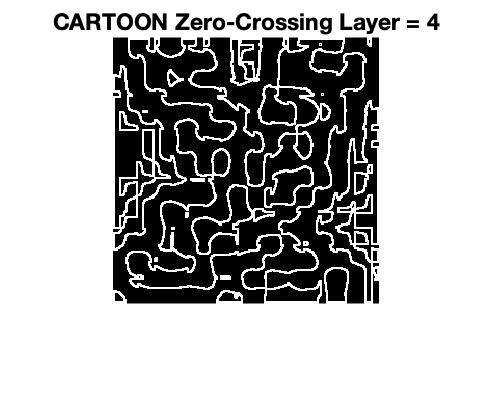
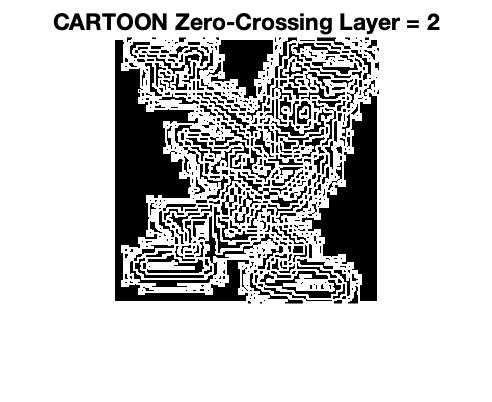
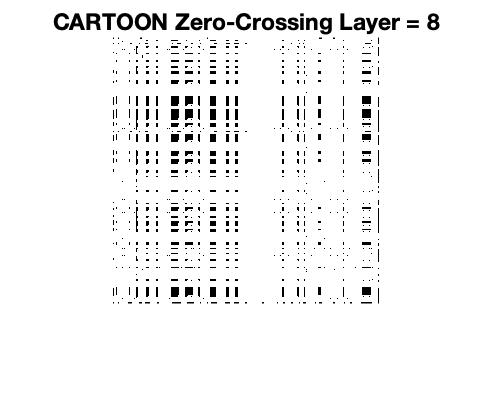




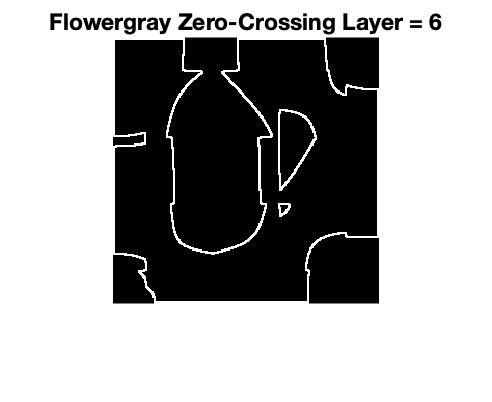
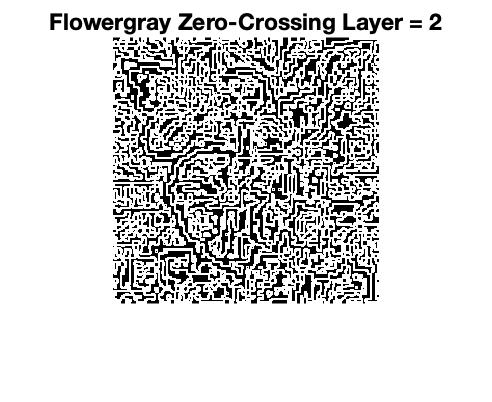
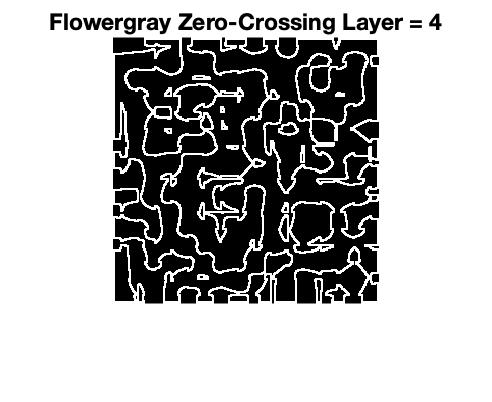
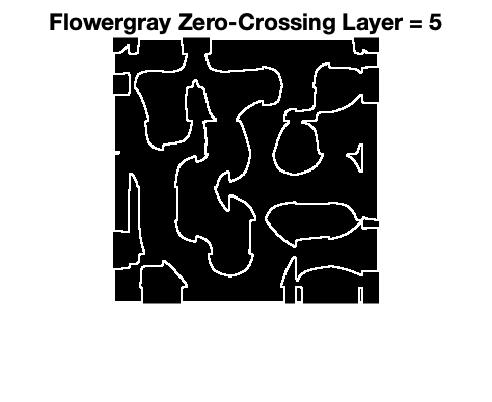
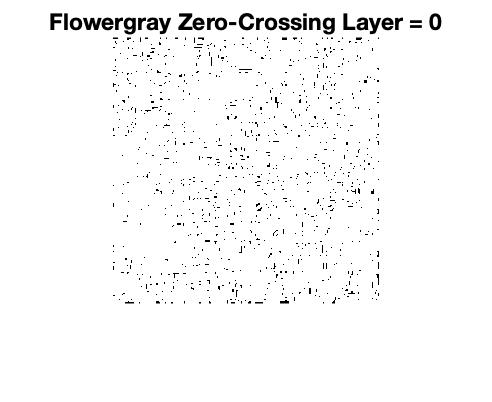
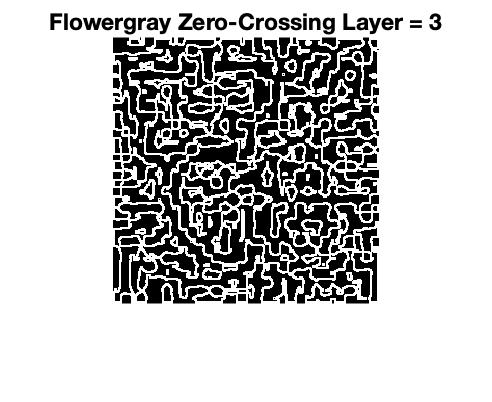
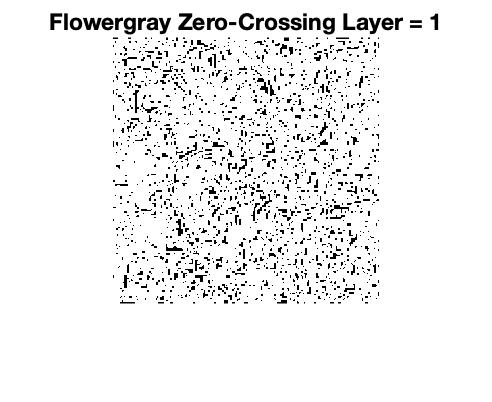
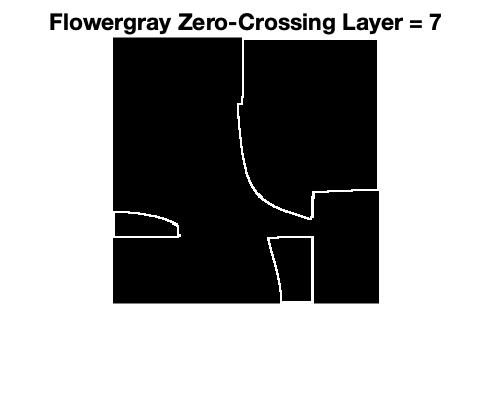
Detect the zero crossing in the segmented image. This is done by tagging any pixel which has at least one neighbor who is of different value than the pixel itself.

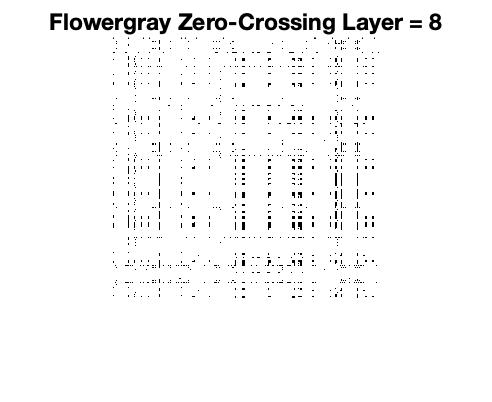
Cartoon

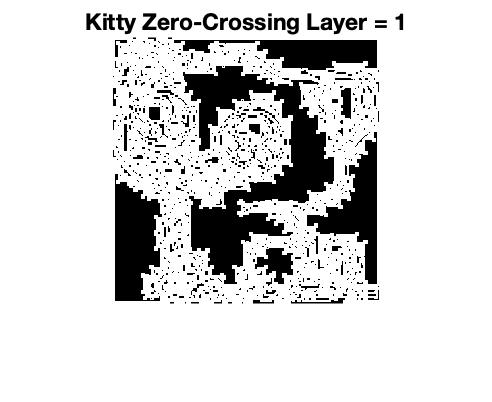


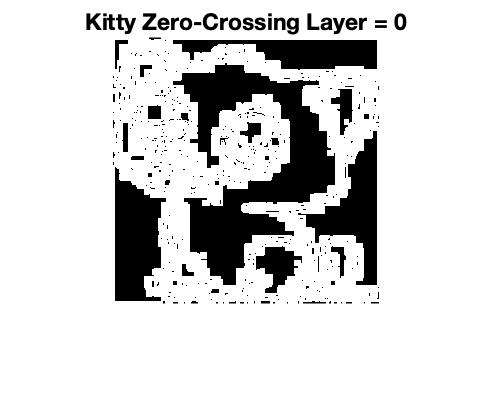


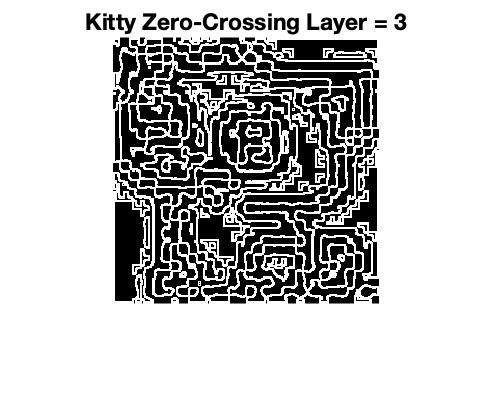
Flower

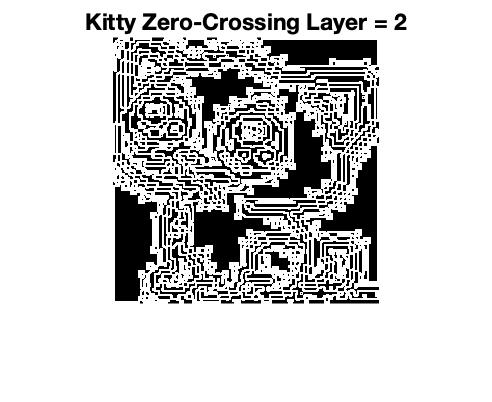


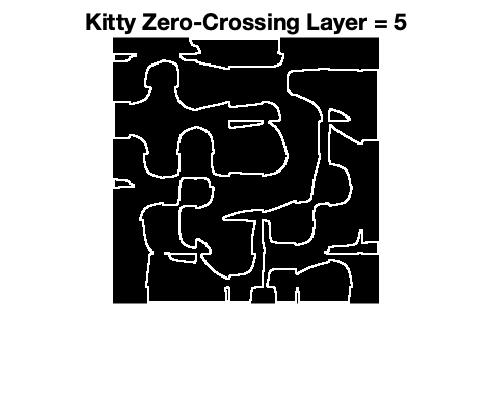


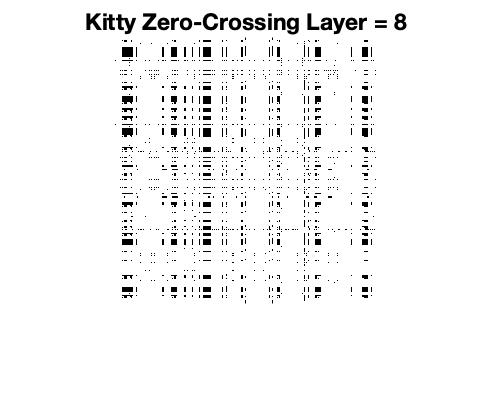
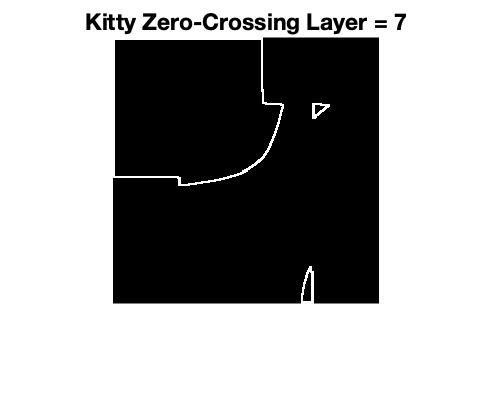
Kitty

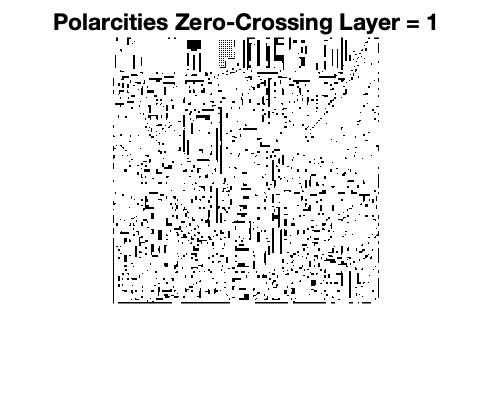
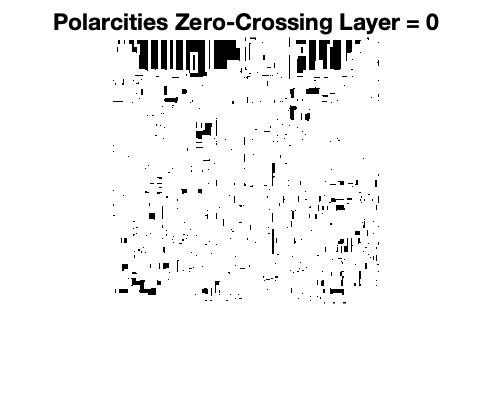


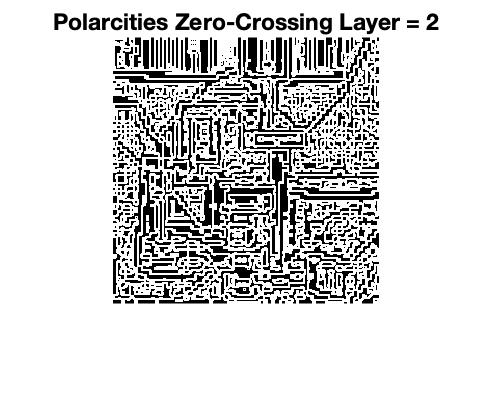
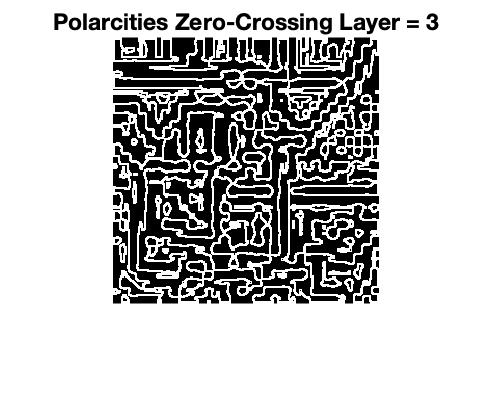


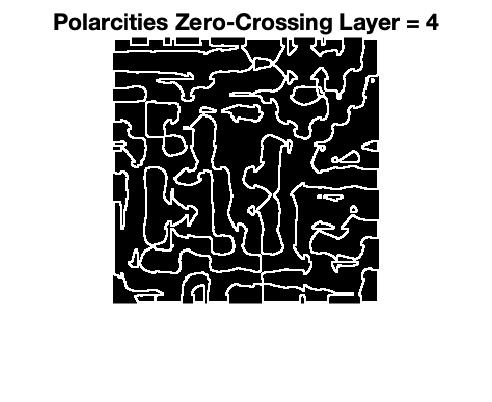
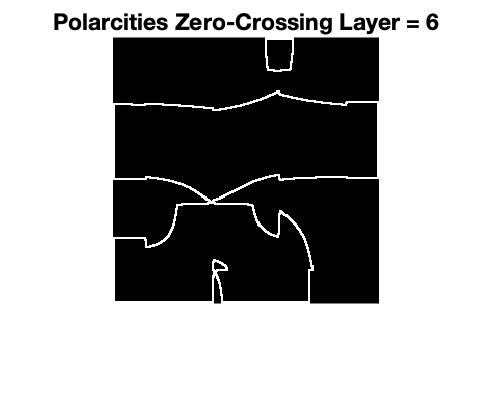
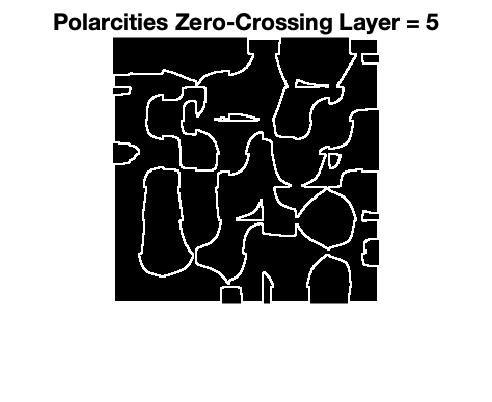
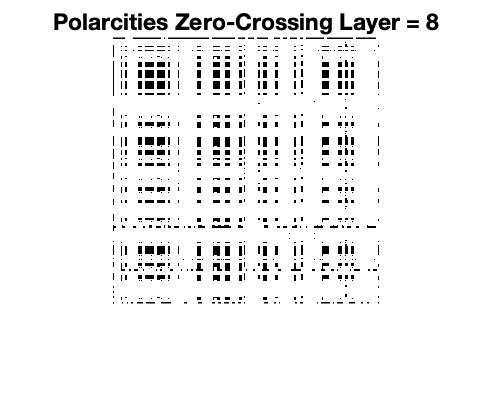




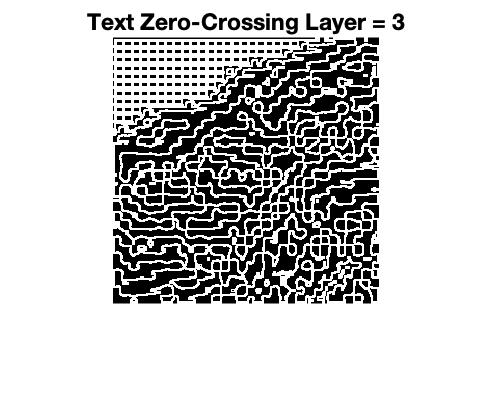
Polarcities

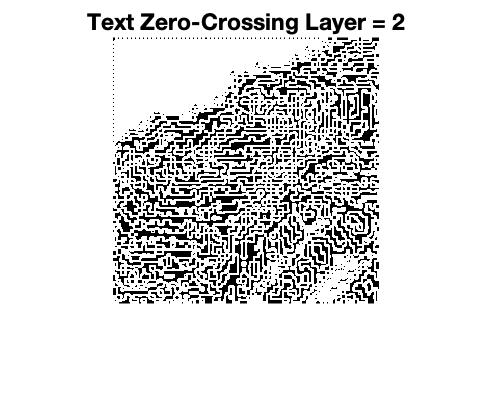
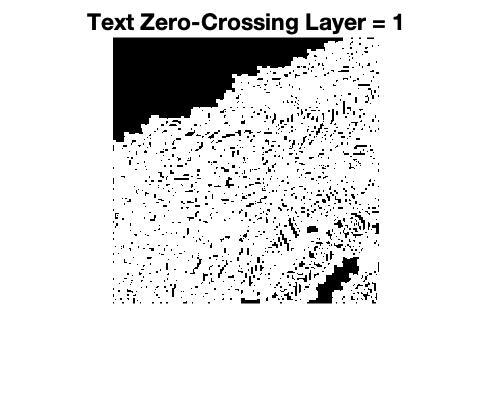
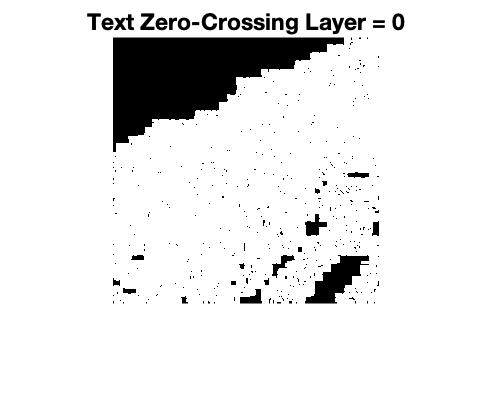


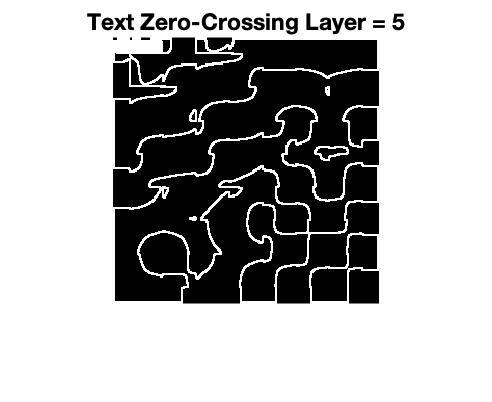


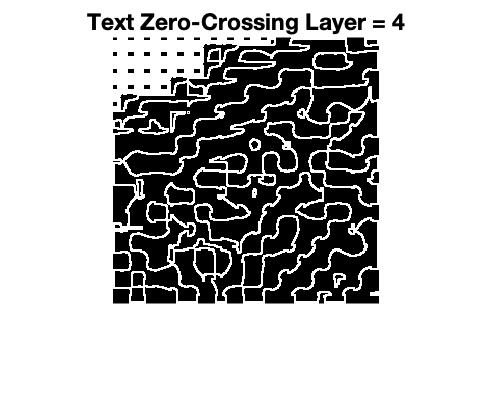
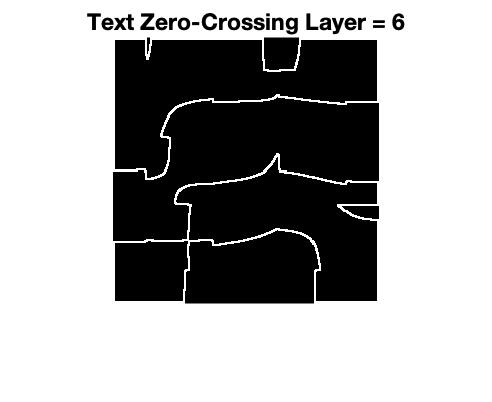


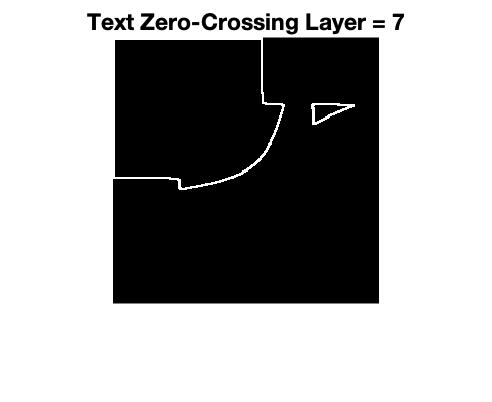
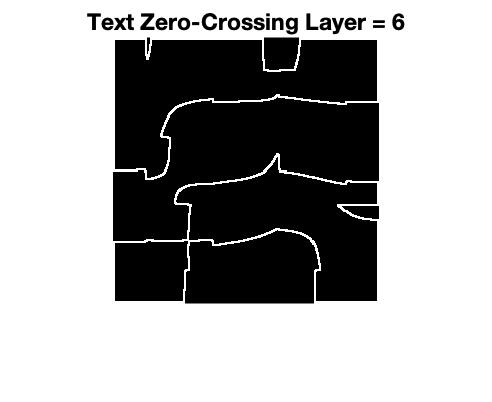
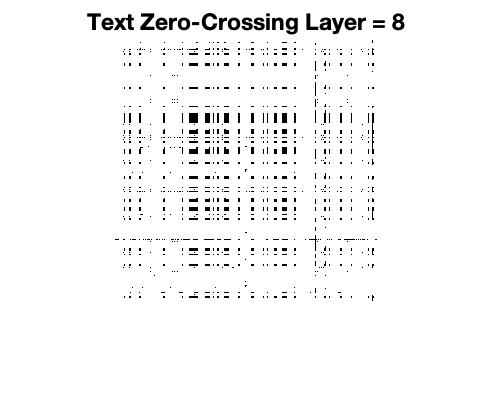
Text





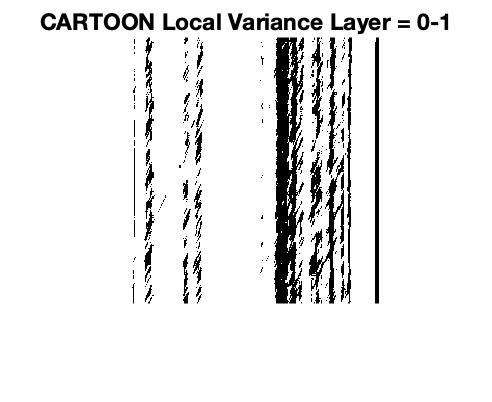
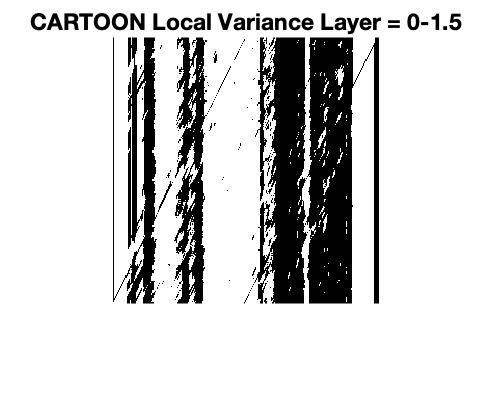


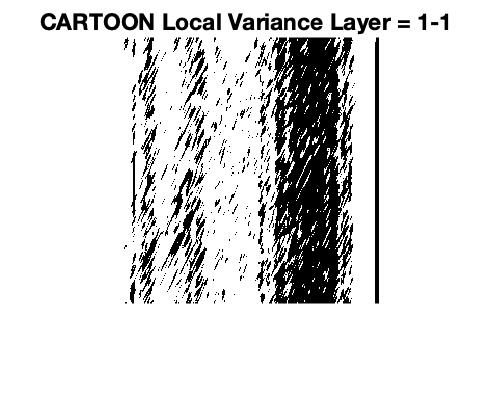
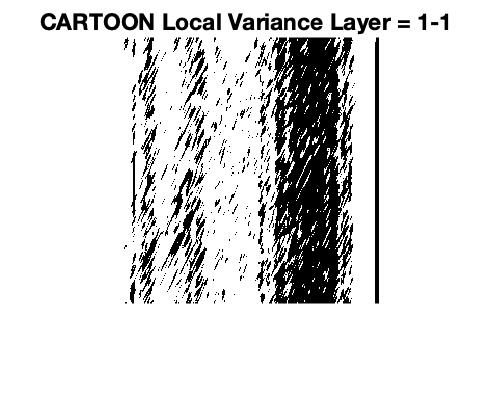
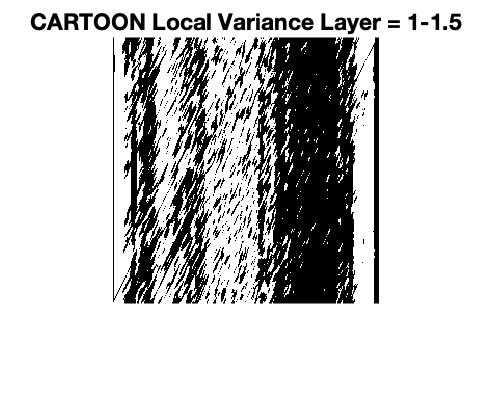
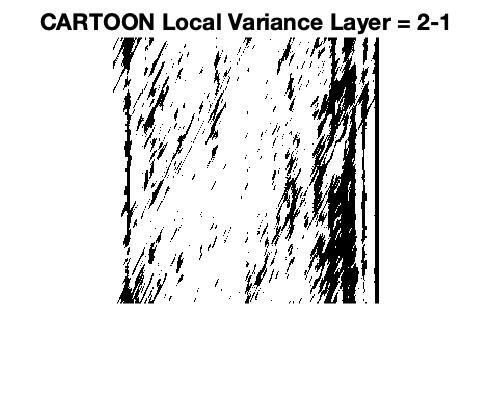
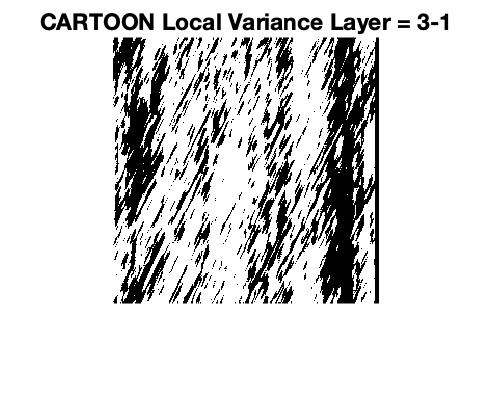
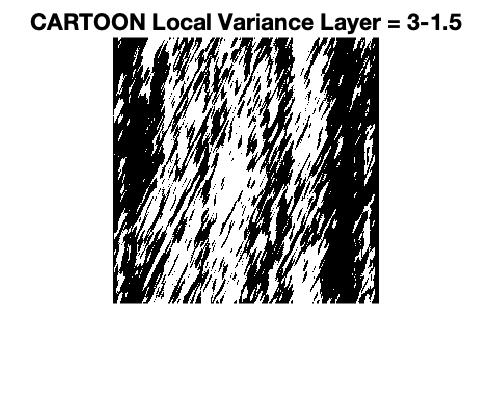
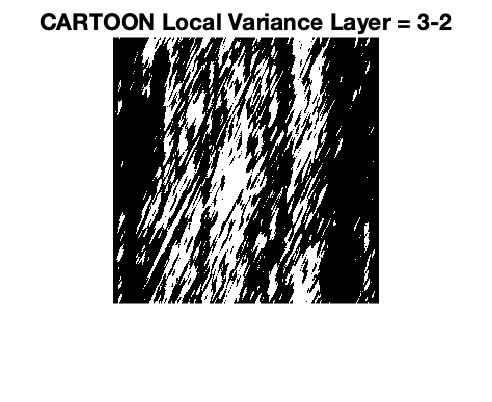
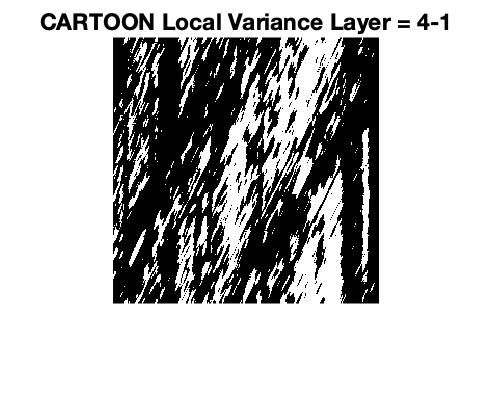
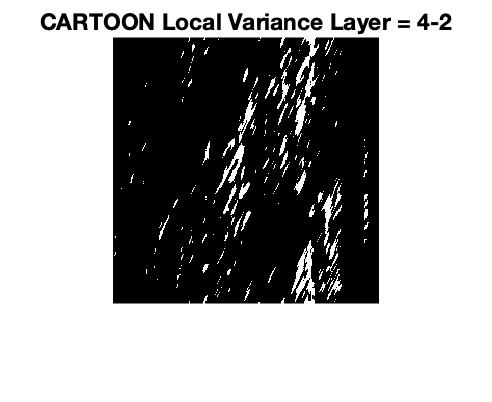
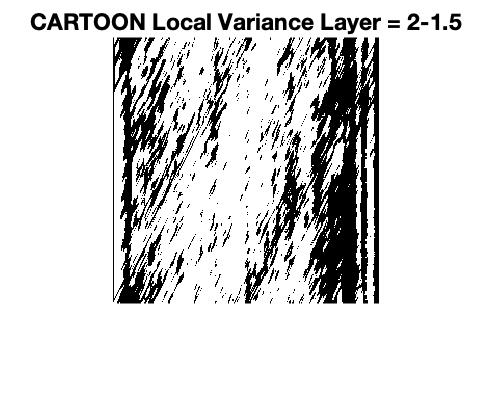
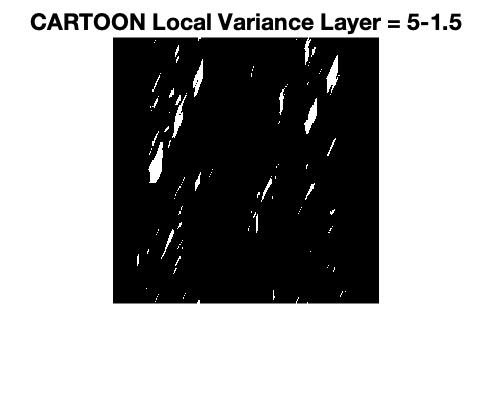
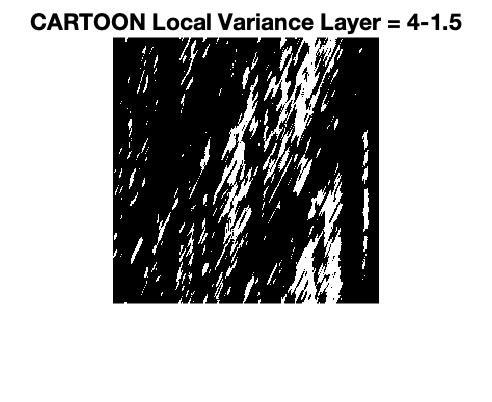
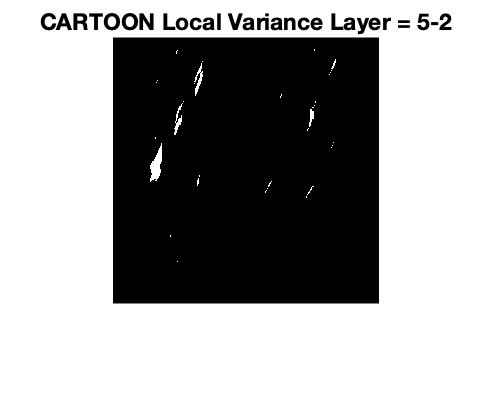


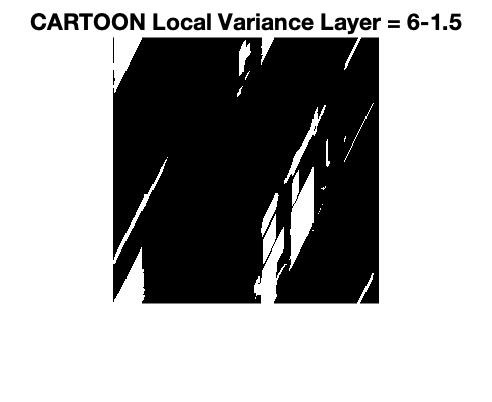
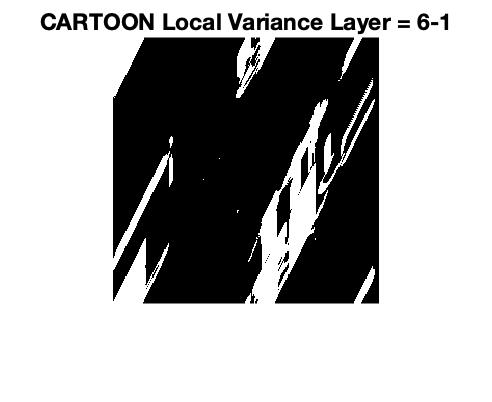
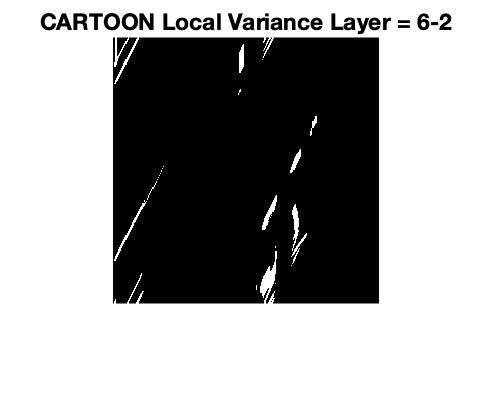
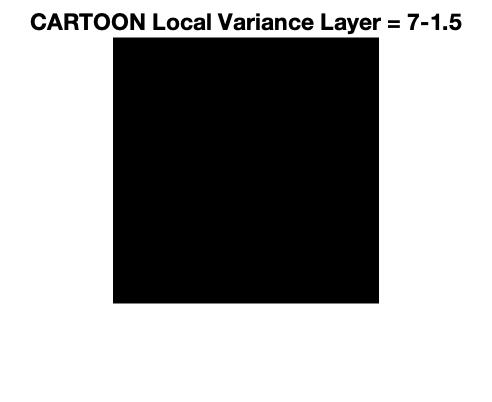
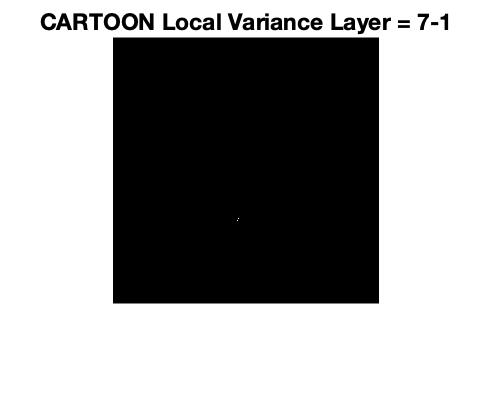
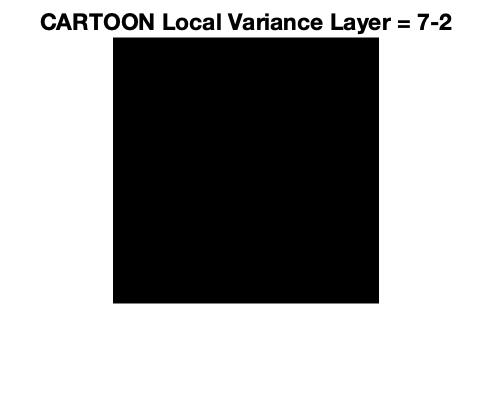
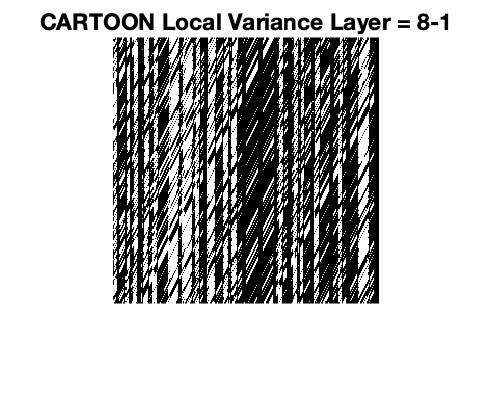
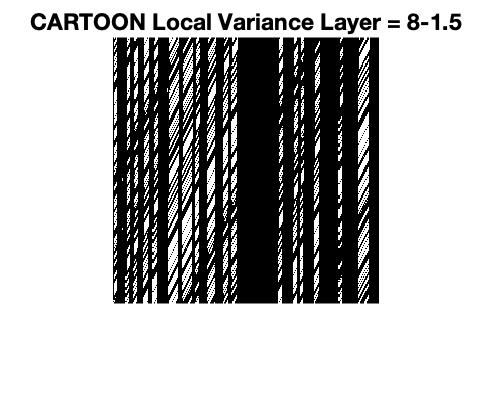
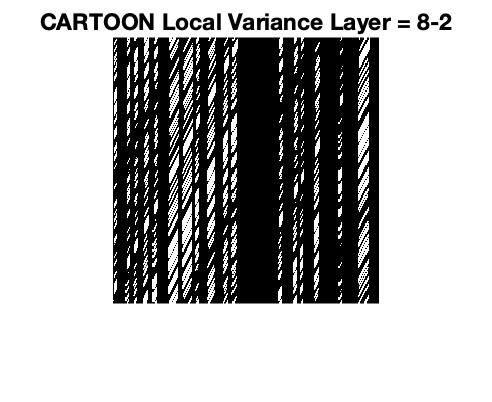
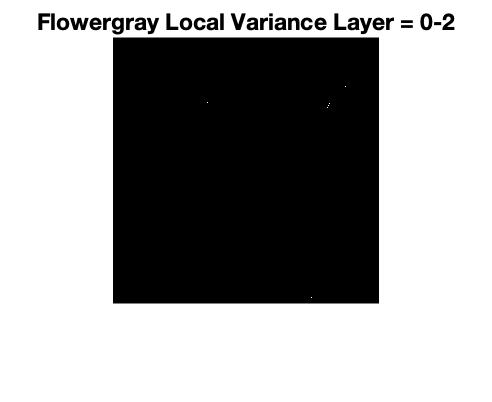
Step 4:

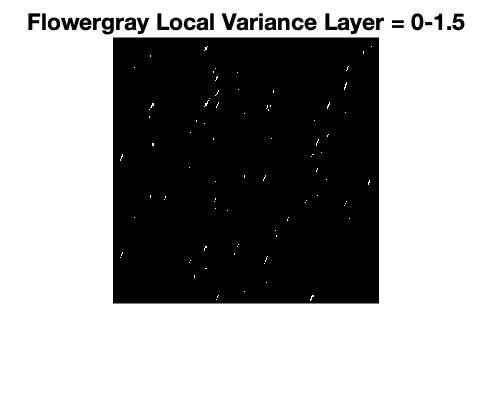
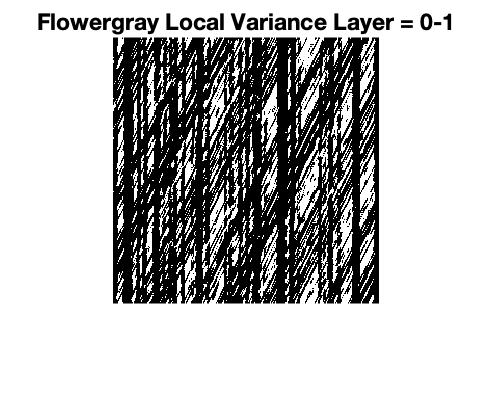
Examine the pixels surrounding the zero crossing pixels in the second order derivative image. Calculate the local variance and mark it as an edge pixel if this value is greater than a certain threshold.

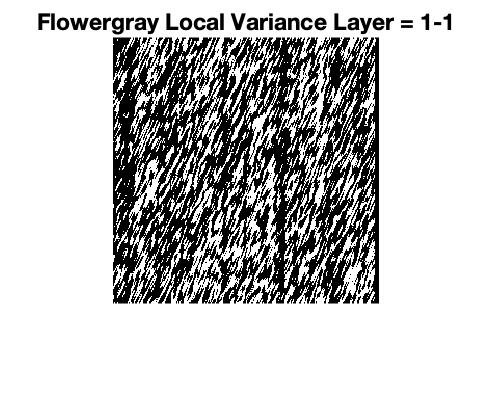
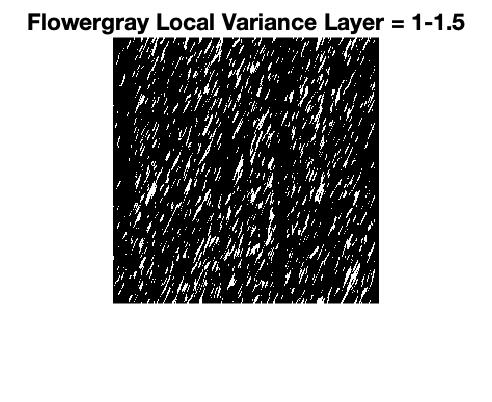
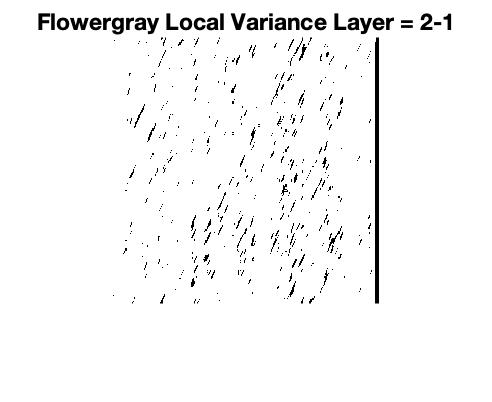
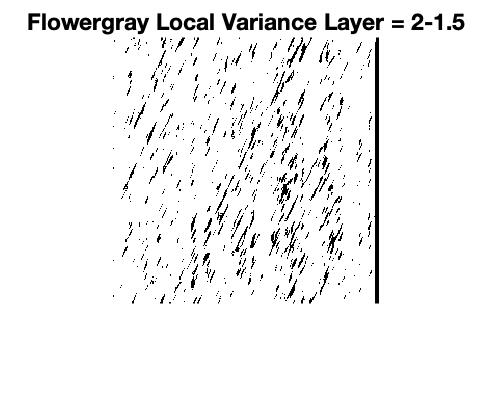
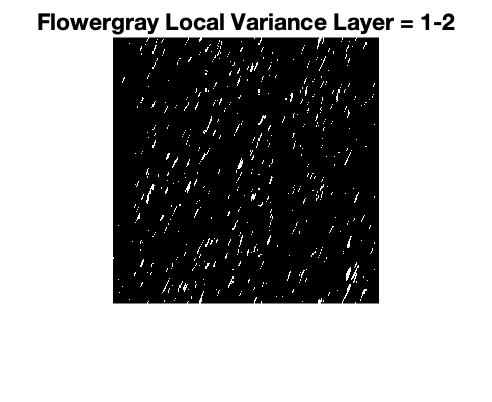
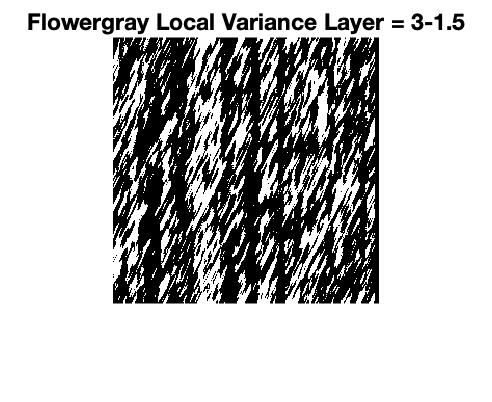
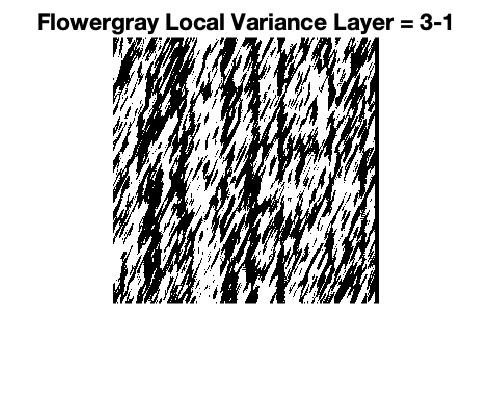
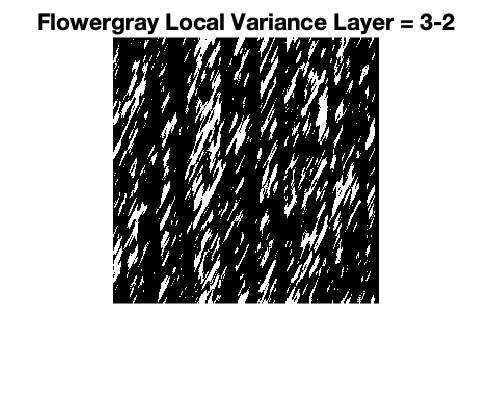
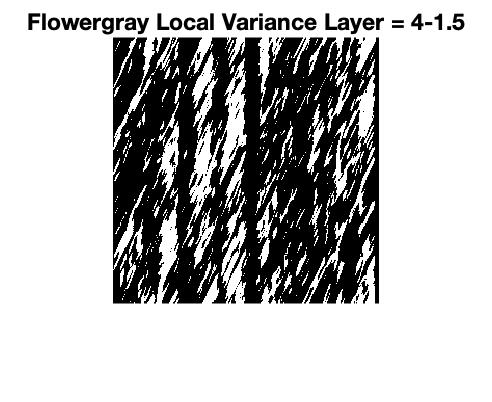
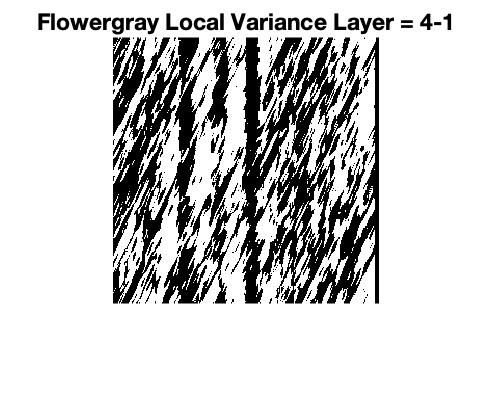
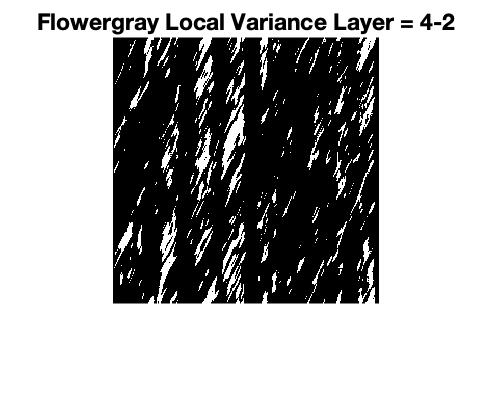
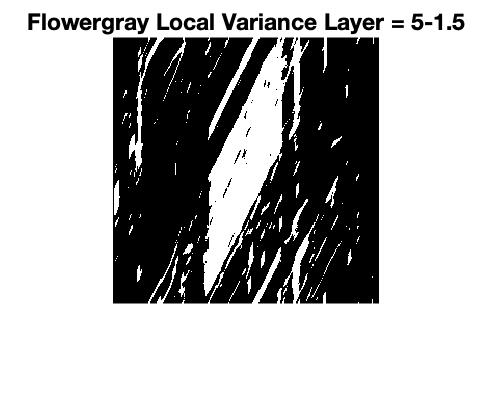
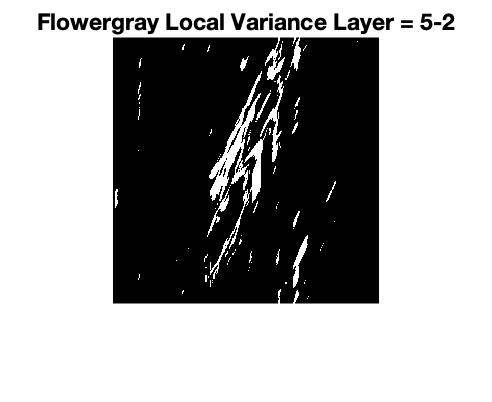
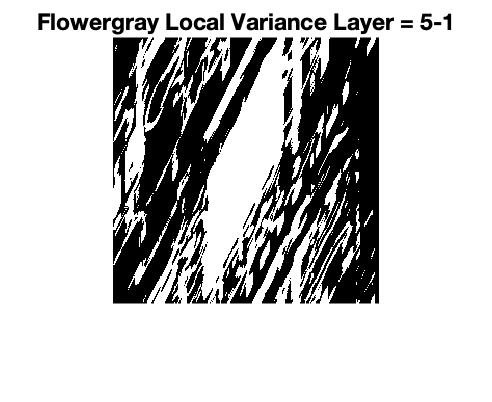
Cartoon

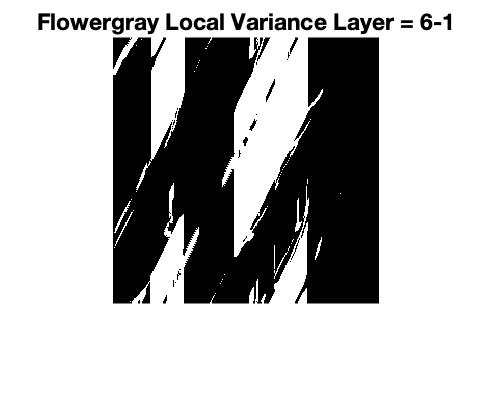
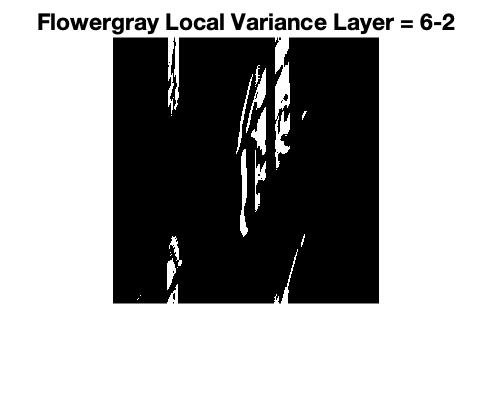
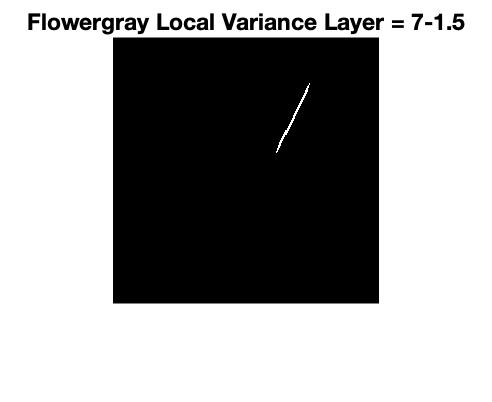
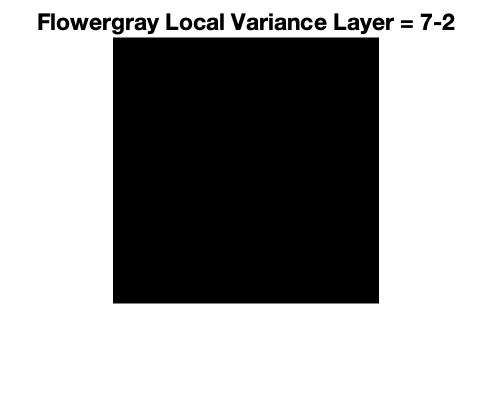
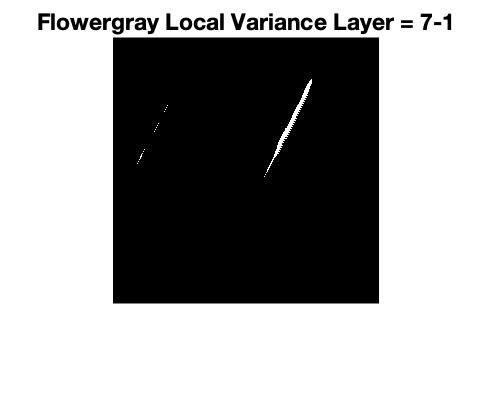
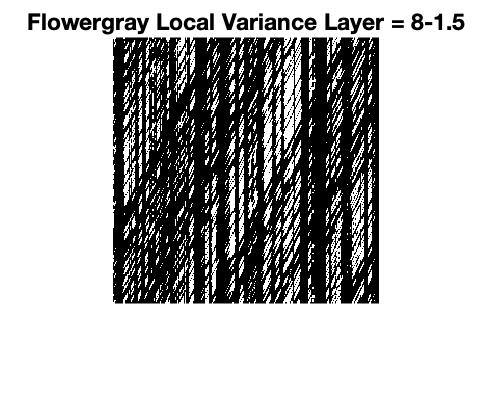
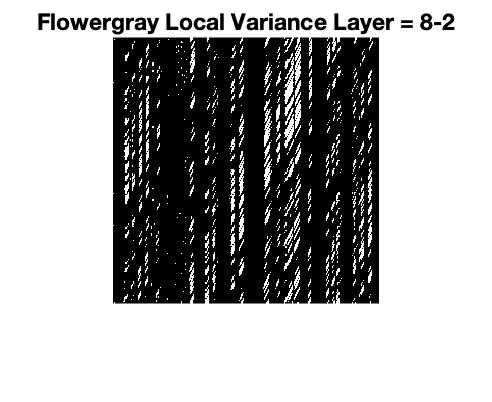




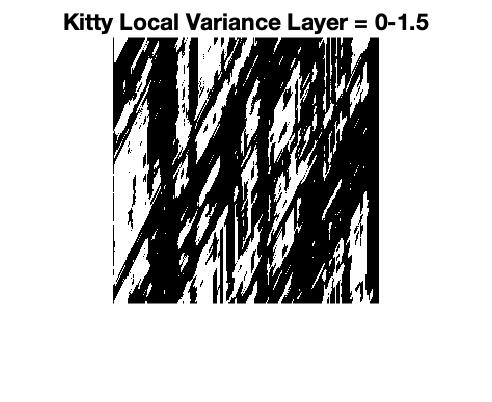
Flower

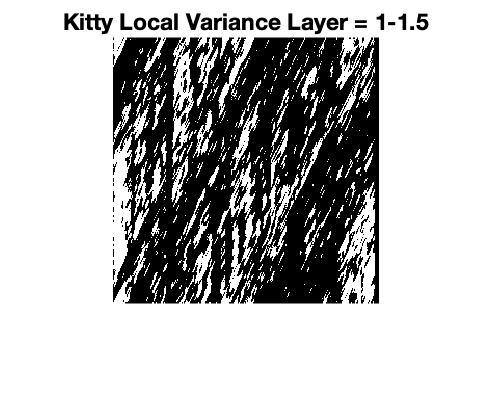
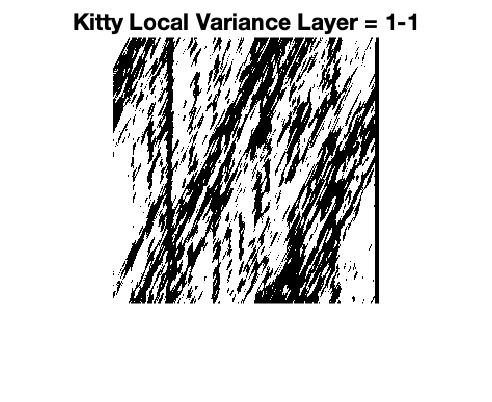
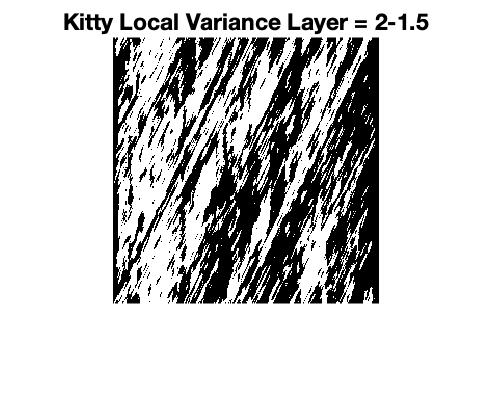
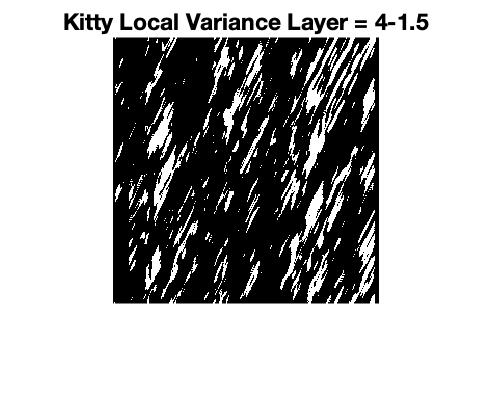






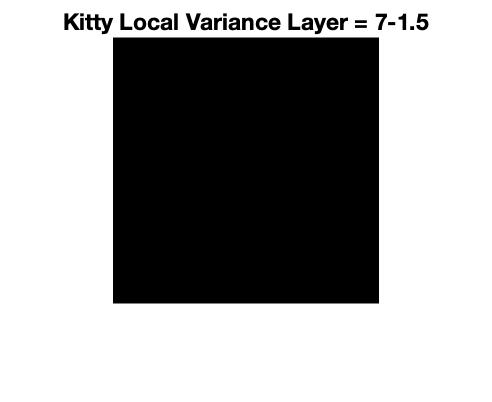
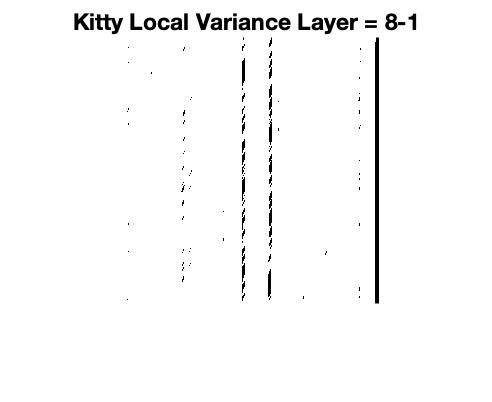
Kitty











Polarcities

Text

**Multi-Resolution Spline**

* Choose 3 pairs of images from the gallery. Show the original images, your mask and the final result in your PDF file