CS6640 A6
Cade Parkison
11/14/2018

Assignment 6: Segmentation: Snakes and Level Sets

Note: The test code used to generate these results can be found in the file testingA6.m. The file A6_testing.ipnb is a jupyter notebook file I used for the rapid testing of all functions and tests.

Problem 1: Active Contours

The objective of this problem was to improve the segmentation of moving objects by applying an active contour function to our results from previous assignments. This was made easier by the use of a matlab built-in function called activecontour(). This function takes in an image, a mask, a max number of iterations, and some optional parameters. First, my CS6640_ac function loops over each frame in the video Object and binary mask of objects in the M object. I then use Matlab's bwlabel function to find the connected region with the largest size and discard any other segmented regions in the binary segmentation movie. This could easily be adapted to work on all segmented objects, I just limited it to the largest region for sake of simplicity and to speed up run time. Next, I passed each binary mask and each original video frame to the activecontour function. The output of this function is another binary mask, this time of a hopefully smaller region surrounding the moving object. This binary image was then stored in the output array im_seg.

The results of this investigation were promising, although I found the computation time needed for Matlab's activecontour function to be very high. In my experimentation, I played around with various parameter settings for this function, including the maximum number of iterations and the active contour method used for segmentation. The two options for the method were 'edge' or 'Chan-Vese'. The activecontour function also can take in either a binary image of the edge pixels or a binary image of the entire segmented region. I opted to use the latter, as the Movie structures 'M' were already in that form. The function uses the boundaries of this object mask to define the initial contour position used for the contour evolution to segment the image.

Below is a summary of my results. First, I tried the activecontours function with a maximum iteration of 150, and an active contour method of 'edge'. This took approximately 15 minutes to compute for the entire video3.avi file. As can be seen below, the function only ran active contours on the largest of the moving object, which was the moving car. The left image in Figure 1 is from the original results of Assignment 4, which was used as an input to CS6640_ac. The right image is the result of running active contours. As can be seen, the segmented region is markedly smaller and closer to the boundary of the car.

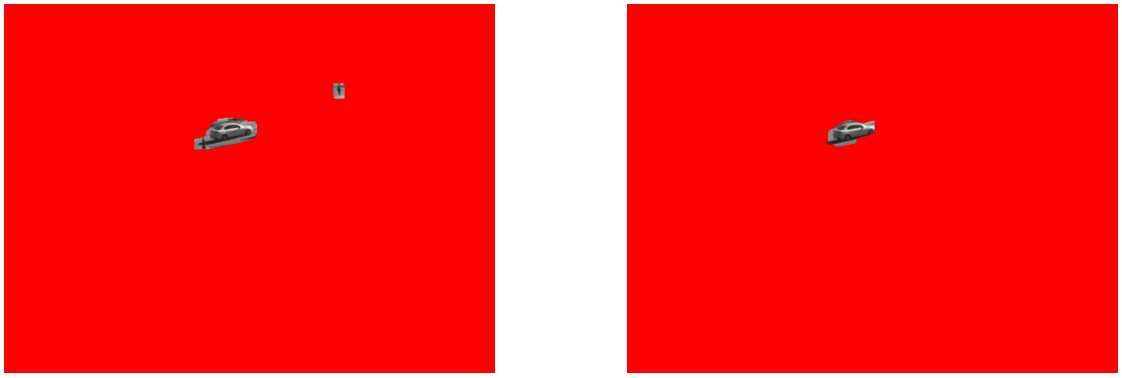

Figure 1: Active Contours, frame 51, iterations=150, method=edge

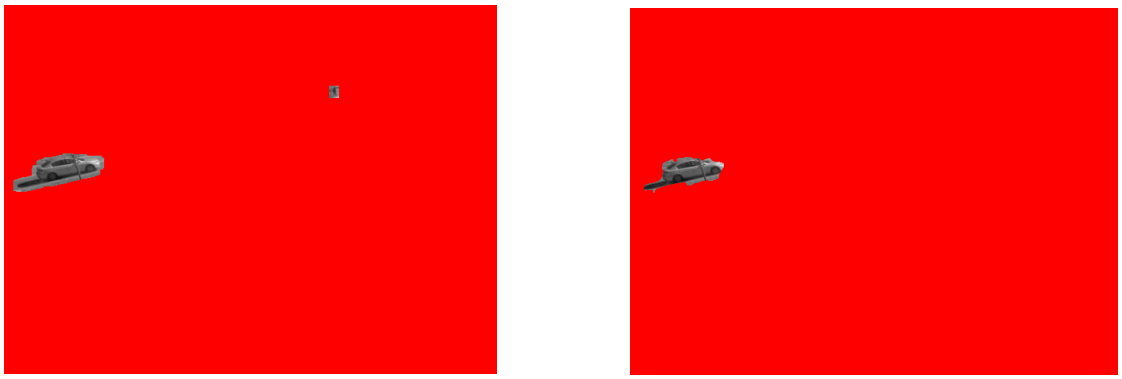

Figure 2: Active Contours, frame 15, iterations=150, method=edge

I then ran the same tests as above, but with a maximum iteration of 100 instead of 150. This substantially reduced the computation time while hardly effecting the quality of the results. These results can be seen below for frames 15 and 51 of video 3, left and right respectively.

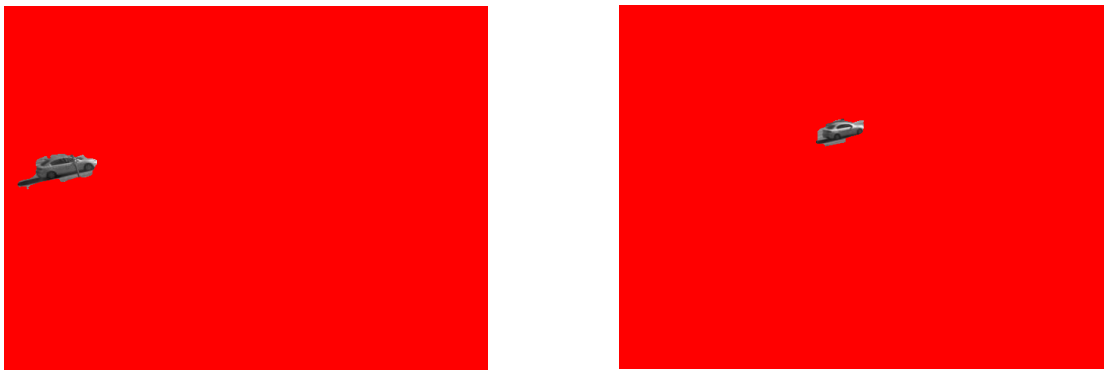

Figure 3: Active Contours, frames 15 and 51, iterations=100, method=edge

Finally, I experimented with the Chan Vese method for the active contours function. This turned out to give less accurate results, even expanding the contours to outside the initial binary mask region. This method did compute much faster than the edge method. Interestingly, this method did seem to be well suited for identifying the trees in front of the moving car. Maybe there could be a future use case for this type of segmentation. The results for frames 15 and 51 are showed below.

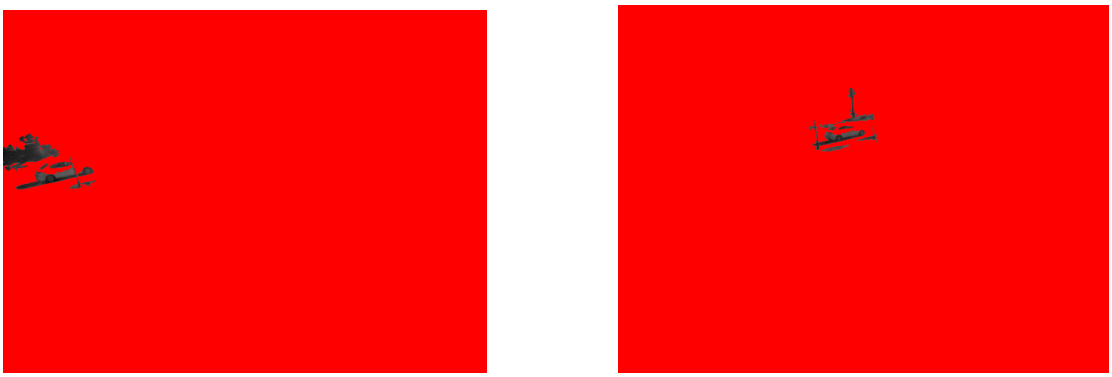

Figure 4: Active Contours, frames 15 and 51, iterations=100, method=Chan-Vese

Overall, active contours using the edge method proved to increase the accuracy of our segmented regions around moving objects, albeit only slightly. There are still parts of the segmented regions that do not represent the moving object itself. I believe through more refinement this could be improved by further tweaking the active contours function parameters or even writing an active contour function myself.

Problem 2: Level Sets

In this problem, I explored level sets as a means of providing better segmentation on a moving object. The Level Set Method looks at an image as a topology, and uses the evolution of a level set curve to segment pixels in regions with similar intensity differences. To do this, our function computes the magnitude of pixel gradients, and uses these magnitudes to determine if a pixel should join the level set region or not. As the boundary of the zero level set region expands, new pixels are included in the segmented object if the difference in gradient magnitude is not too large between them. This has the effect of stopping the expansion of the level set curve along sharp edges in the image.

In my implementation, I set the initial radius of the zero level set to 4 pixels. I then tested my function on frame 51 of video 3 in various locations in the image. One issue I ran into, due to the formulation of the directions of propagation variables, was I had to limit the bounds of my for loops by one pixel on either size. For an MxN image, my function only computes phi values for (M-2)x(N-2) pixels. This could possibly be improved in the future by padding the original image along the borders.

For my first test, I set the initial zero level set to pixel location (425,500), which corresponds to the sidewalk in the bottom right of frame 51 and can be seen below.

Figure 5: Original Image

The results of CS6640_level_set for this test are summarized below. My function has a switch called DISPLAY, which if true will display the evolution of the level sets as a sequence of images. This switch is false by default.

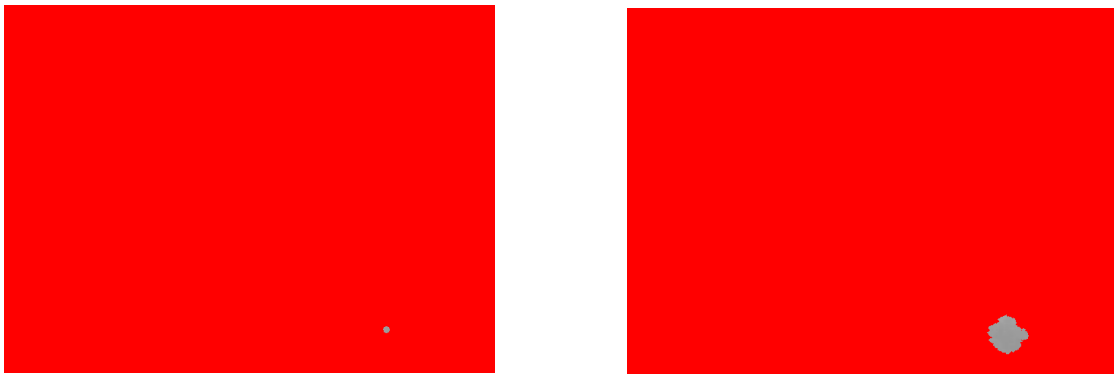

Figure 6: Iterations 1 (left) and 50 (right)

Figure 7: Iterations 100 (left) and 200 (right)

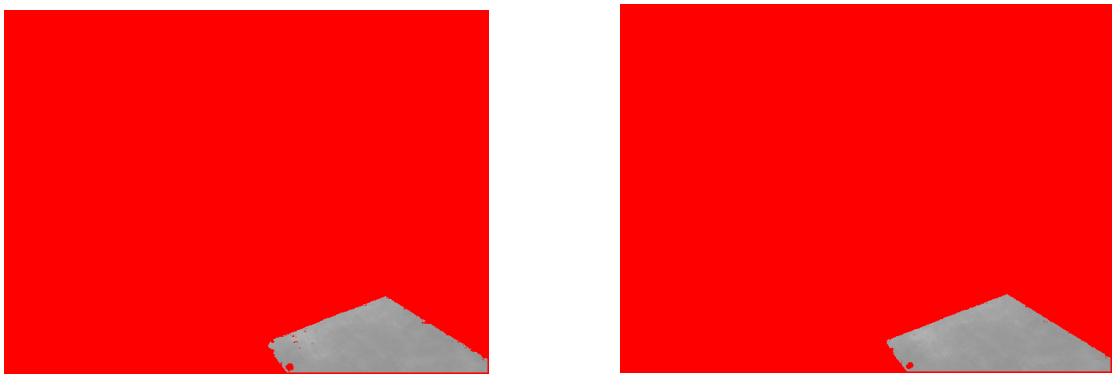

Figure 8: Iterations 500 (left) and 1000 (right)

I ran the algorithm for 100 iterations and a delta_t of 0.8, and the results were very promising. As can be seen above, the level set evolution grew until it reached the borders of the concrete tile. There was very

little change from iteration 500 to 1000. The plot of the change in phi for each iteration can be seen below. As can be seen, the difference in each successive phi value gets smaller as the iterations increase.

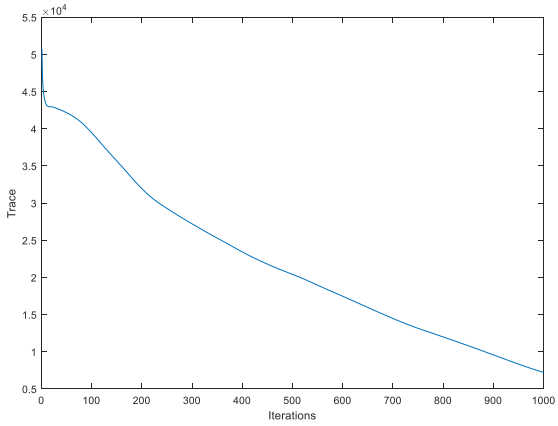

Figure 9: Plot of phi differences between iterations

Next, I tried the same test as above, but on the hood of a car in the parking lot, as seen below.

Figure 10: Starting point on hood of car

Due to the larger differences in gradient magnitudes in this location, the expansion wave moved much slower, but in the in, it did a pretty good job of accurately segmenting the hood of the car. Below is the result after 1000 iterations, where the red pixels represent phi values of less than or equal to zero, which is the inverse of the images above.

Figure 11: Segmented pixels with Phi<=0

The same tests as above were done on a couple more locations in the original image, results are below. A starting radius of 2 and a max iteration of 1000 was used for both.

Figure 12: Parking Lot: Starting point (left) and final segmented region (right)

Figure 13: Side Panel of moving car: Starting point and final region

As can be seen above, this algorithm proved very useful results. It was best suited for large regions of similar pixels, like the sidewalk and parking lots. In these cases, the expanding level set wave very quickly found pixels nearby with similar changes in gradients. This could prove very useful for future image classification algorithms. In the case of the moving car, the expanding level set could not move very far in 1000 iterations, although it did a fairly good job of segmenting pixels belonging to the side panel of the car. I could see potential in future versions of this algorithm where this function was called on many different parts of the car to try and get useful information about the entire car.

Curiously, I was unable to achieve the results shown in class, where the expanding level set wave eventually explored all regions of an image. My implementation seemed to always get stuck in wells with high gradient magnitudes stopping it from expanding further. This was best demonstrated in the first test on the sidewalk in the foreground of the image.

Overall, I found this method to be very promising for image segmentation and classification. The algorithm runs fast and easily finds similar pixel regions. One downside to this method of segmentation is that the starting point must manually be set for each test. A potential improvement could be made by having a higher level function that calls this function many times across different parts of an image and then combines each region into a single image with multiple segmented regions.