# Computer Visual Homework 4

ID:113598045　Name:林翡

- Explain your code:

1. **Convert to grayscale image and Gaussian Filter (convertGray、padding、Gaussian_kernel、Gaussian_filter Functions)**

   These functions are the same as those in homework 3.

2. **Image Pre-processing (preprocess_image Function):**

   In the preprocess_image function, the goal is to pre-process the input image. First, convert the image into a grayscale image, and then perform Gaussian filtering [Lines 81-83]. Use the Sobel operator to calculate the image gradients Gx and Gy [Line 85-86]. Calculate the gradient magnitude, and normalize the gradient magnitude to the range of 0~255 [Lines 88-89].

3. **Set Initial Contour (initialize_contour_ellipse、initialize_contour_rec Function):**

   In the initialize_contour_ellipse function, the goal is to initialize the contour of an elliptical shape on the image (Used in img1.png and img2.png). Use the parameters ellipse_scale_x, ellipse_scale_y to adjust the relative size of the ellipse. First, get the image dimensions and calculate the center point of the image [Lines 95-100]. Calculate the length of the horizontal and vertical semi-axes of the ellipse [Lines 102-103]. Generate a total of 70 points on the ellipse, and add the calculated points to the initial_contour list [Lines 105-110].

   In the initialize_contour_rec function, the goal is to initialize the contour of a rectangle shape on the image (Used in img3.png). Similar to the initialize_contour_ellipse function, use the parameters rect_scale_x, rect_scale_y to adjust the relative size of the rectangle. Similarly, first calculate the center point of the image, and then calculate the width and height of the rectangle [Lines 116-124]. Evenly distribute the points to the four sides of the rectangle [Line 126]. Generate points on the top, right, bottom, and left edges of the rectangle [Lines 127-147].

4. **Active Contour (calculate_energy_components、active_contour_step、active_contour Functions):**

   In the calculate_energy_components function, the goal is to calculate three energy components in the active contour model: continuity energy, curvature energy, and image energy. First, define the function that calculates the Euclidean distance [Lines 151-152]. Calculate continuity energy (Econt) [Line 154]. For the part that calculates the curvature energy (Ecurv), mid_point is curr_point, which is assumed to be the center point of symmetry between the previous point and the next point [Lines 156-158], and then calculates the distance between next_point and mid_point [Line 159]. If next_point does not exist, the curvature energy is set to 0 [Lines 160-161]. To calculate the image energy (Eimg), first ensure that the coordinates (x, y) of the current point do not exceed the boundary of the image. If the point is within the range, take the gradient value of the point in the magnitude image and take the negative value as the energy [Lines 165-166]. Otherwise, the point is out of range, set the image energy to 'inf' [Lines 167-168].

In the active_contour_step function, the goal is to adjust the position of each point within a certain range and update the contour by calculating the energy total of the contour points. Initialize the new_points list for updated contours [Line 173]. Use a for loop to traverse each point in the contour [Lines 174], prev_point is the previous point of the current point,next_point is the next point of the current point, and it loops back to the starting point if the current point is the last point [Lines175-176]. Initialize the minimum energy Emin and the best position best_point [Lines 177-178]. Use nested loops to search for candidate points within the range [Lines 179-181]. Calculate the energy and total energy of candidate points [Lines 183-184]. Update the best position. If the total energy of the current candidate point is less than the lowest energy currently recorded, update Emin to the current lowest energy value and best_point to the current best position. If the energy is less than threshold, end the search early [Lines 185-189]. Store the best position found in new_points [Line 190].

In the active_contour function, First, initialize the contour points 'points', the energy change value 'energy_change', the current iteration number 'iteration', and store the contour list 'contours_video' after each update [Lines 196-199]. When the maximum number of iterations 'max_iterations' is not reached and the energy change 'energy_change' does not converge to the specified threshold 'convergence_threshold', call active_contour_step() to update the contour [Lines 200-201]. Calculate the energy change by summing the distances of all points and taking the average [Line 203]. Overwrite the current contour points 'points' with the updated contour points 'new_points' and enter the next iteration [Line 204]. Draw the updated contour [Lines 206-209]. If the energy change is less than ‘convergence_threshold’, the loop is stopped and the contour is considered to have converged [Line 210-211]. Each time a round of updates is completed, the iteration count is increased by 1 [Lines 212].

5. **Save the result videos, measure the number of iteration and DSC value(calculate_dsc、save_video、save_image_with_contour Functions):**

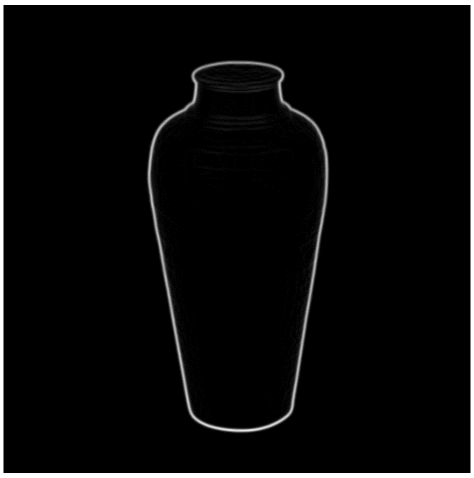
   In the calculate_dsc function, calculate according to the DSC formula. First calculate the intersection of the two binary masks ground_truth_mask and predicted_segmentation_mask [Line 217]. Then calculate the union part, then calculate the union part, and finally return the size of the intersection part multiplied by 2, and then divided by the size of the union part [Line 218-219].

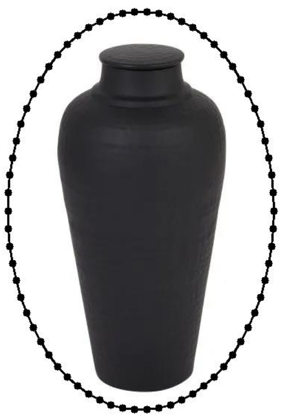
   In the save_video and save_image_with_contour functions, they save the gradually updated contour points into video and image types respectively.

6. **Main Function:**

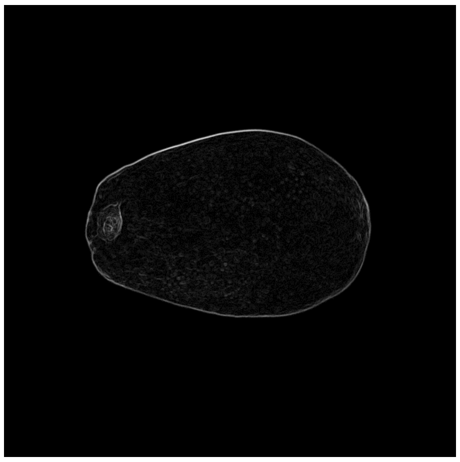
   First, read three input images [Lines 239-241]. Call preprocess_image() to do image preprocessing and store the gradient magnitude image [Lines 243-248]. Call initialize_contour_ellipse() to set the initial contours of img1 and img2, call initialize_contour_rec() to set the initial contour of img3, and store them [Lines 250-255]. Call active_contour() to implement the active contour algorithm and store the final contour image [Lines 257-265]. Create ground truth and the predicted segmentation mask (active contour mask) [Lines 267-276]. Call calculate_dsc() to calculate DSC [Lines 278-280]. Save the video and print the iteration number and DSC value [Lines 282-293].

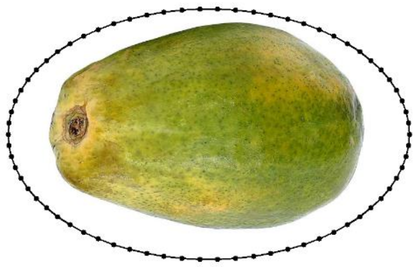
- Please paste 9 output images in your report, the number of iterations, and their respective DSC values for each contour's images.

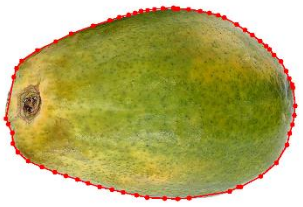

img1_q1.png

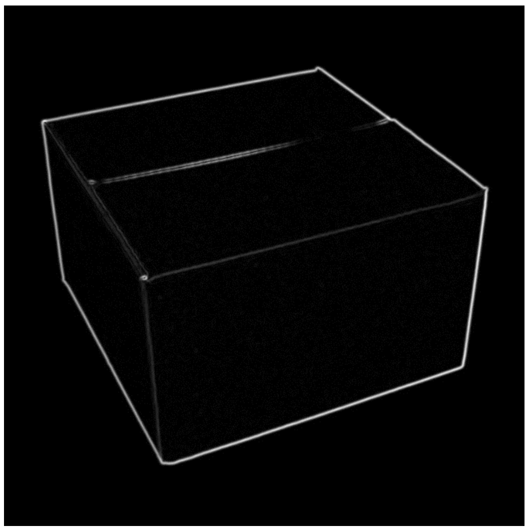

img1_q2.png

img1_q3.png

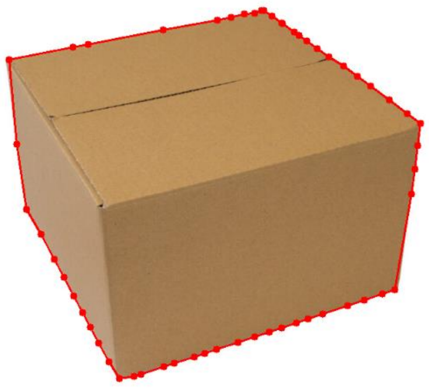

img2_q1.png


img_q2.png

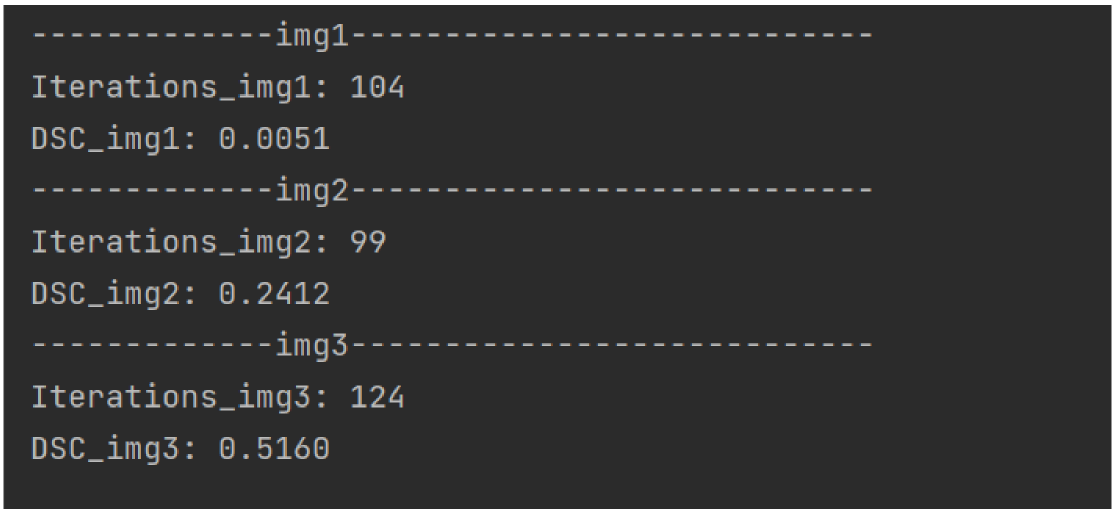

img_q3.png


img3_q1.png

img3_q2.png



img3_q3.png



```
-------------img1----------------------------
Iterations_img1: 104
DSC_img1: 0.0051
-------------img2----------------------------
Iterations_img2: 99
DSC_img2: 0.2412
-------------img3----------------------------
Iterations_img3: 124
DSC_img3: 0.5160
```

The number of iterations and corresponding DSC values of the three images

- Please state in your report the parameter setting that you choose for each input images.

  Regarding img1.png, the main focus has been on adjusting the search range for candidate points in the active_contour_step() function, specifically the offset range (gx, gy). When the range for x is set too small, the points fail to approach the edges. Additionally, it was observed that if min_range_y is set to less than -1, most points tend to move away from the edges of the vase (with other parameters unchanged). After multiple adjustments, setting the range for gx to -15 to 2 (exclusive) and gy to -1 to 17 (exclusive) results in the points being closest to the edges.

  Furthermore, I adjusted alpha to 1.5, beta to 1.5, and gamma to 2.0. When alpha and beta are less than 1.5, most points also move away from the edges. Similarly, when gamma is less than 2, the concave area in the upper-right corner of the vase does not fit closely enough. These adjustments and comparisons were made while keeping other parameters unchanged. The maximum number of iterations (`max_iterations`) is set to 200, and the convergence threshold (`convergence_threshold`) is set to 0.85.

For img2.png, the adjustment also focused on the search range for gx and gy, but this time the range was set smaller (gx is -4 to 4 (exclusive)，gy is -1 to 5 (exclusive)). If the range is set as large as in img1.png, the contour points in the upper-left corner of the fruit fail to align with the edges.

The parameters max_iterations, convergence_threshold, alpha, beta, and gamma were not specifically adjusted and remain the same as in img1.png. However, img2.png is relatively easier to adjust compared to img1.png, as simply modifying the search range nearly achieves perfect edge alignment.

For img3.png, since it is a box, the initial points were set in a rectangular shape. The search range for gx and gy also needed to be expanded. If the range was set too small, the upper-right edge of the box would fail to align. This edge was the most challenging to adjust. The search ranges for gx and gy were set to -18 to 4 (exclusive) and -1 to 14 (exclusive), respectively. It can be observed that for all three images, the min_range_y of gy could not be less than -1.

Additionally, the convergence_threshold was adjusted to 0.75. If it was set to 0.85, the upper-right and lower edges of the box would not align well. The parameters alpha, beta, and gamma were also adjusted to 1.0, 1.0, and 3.0, respectively. Setting alpha and beta too large would cause the upper-right edge of the box to fail to align, while setting gamma too small would result in the lower-left corner points not aligning properly.

Through repeated adjustments and comparisons, these parameter settings produced better contour alignment results.