CALIFORNIA STATE UNIVERSITY, LONG BEACH

College of Engineering
Department of Computer Engineering and Computer Science
Dr. Thinh V. Nguyen
Spring 2007
CECS-553/653: Machine Vision

PROJECT 1

Name: **Foss, Shannon**
Last, First

- **Dates:** Date assigned: Wednesday February 21, 2007. Date due: Wednesday March 14, 2007. Late submissions will receive penalty at 10% per day. This project is worth 15% of the project grade.

- **Objectives**: The objectives of this project includes: (1) to familiarize students with the bitmap image file format, (2) to perform elementary image grey level and multi-resolution operations and histogram equalization, OR (1) to perform research in image thresholding or binarization, and (2) to develop algorithm for binarization using histogram method.

- **Project Description:**

A. Programming Project:

Write a computer program to read in an image named "image.bmp" where *image* is the input to the program and is selected from the image database. Save the image in grey level as "image_grey.bmp" Note that the word "image" should be replaced with the appropriate image file name.

Perform the following operations:

1) Sub-sampling grey level resolutions: display the "image_grey.bmp" image in 64, 32, 16, 4, and 2 grey levels. For the binary image (2 grey levels), the two values should be 0 and 255. Save these images as "image_grey64.bmp", "image_grey32.bmp", "image_grey16.bmp", "image_grey8.bmp", "image_grey4.bmp", and "image_grey2.bmp", corresponding to 64, 32, 16, 8, 4, and 2 grey levels, respectively. Note that these images should have the original spatial resolutions.

2) Sub-sampling spatial resolutions: cut out a sub-image having size of NxN with center at coordinates (x,y) of the "image_grey.bmp" image, where NxN is any one of 16x16, 32x32, 64x64, 128x128. Save this cut-out image as "image_spatialN.bmp", where N is 16, 32, 64, 128. Then, display the image in 16x16, 32x32, 64x64, 128x128, 256x256, and 512x512 spatial resolutions. Save these images as "image_spatial16.bmp", "image_spatial32.bmp", "image_spatial64.bmp", "image_spatial128.bmp", "image_spatial256.bmp", and "image_spatial512.bmp", corresponding the 16x16, 32x32, 64x64, 128x128, 256x256, and 512x512 spatial resolutions, respectively.

3) Histogram equalization. Perform histogram equalization on the "image_grey.bmp" and save it as "image_equalhisto.bmp". Compute the histograms of the "image_grey.bmp" and "image_equalhisto.bmp" and display them as a bitmap image named "image_histograms.bmp" of size 256x512. Put the histogram of the "image_grey.bmp" at the top and the histogram of the "image_equalhisto.bmp". at the bottom for comparison.

Histogram equalization includes the following steps:

- Compute the histogram of the input image h(n) having grey levels from (0, N-1);
- Compute the relative frequency histogram h'(n), defined as the histogram scaled by the image size (xd*yd);

- Determine the mapping of the grey level. The mapping f(n) is incrementally constructed based on the relative frequency histogram h'(n):

$$f(n) = (N-1) \sum_{k=0,n} h(k)$$

- Implementation: h(n), h'(n) should be floating point, f(n) should be integer. The summation of h(k) over k should be rounded up.

A sample code for the histogram equalization is as follows:

```
// compute the histogram h(n)
// max_grey is the maximum grey level, typically 256 for 8-bit grey level
//histo[] and  equal_histo[] are histogram arrays for original and equalized images

for (i=0; i<max_grey; i++){ equal_histo[i]= 0.0; histo[i] = 0.0;}
for (j=0;j<yd;j++)
        for (i=0; i<xd; i++)
                histo[array[i][j]] += 1.0;

// scale histogram to [0, 255] for display purpose
gmin = large_number;
gmax = small_number;
for (i=0; i<max_grey; i++){
        if (histo[i]< gmin) gmin = histo[i];
        if (histo[i]> gmax) gmax = histo[i];
}
gdelta = gmax - gmin;
for (i=0; i <max_grey; i++){
        // scale for display purposes
        histogram[i] = unsigned char ((histo[i] - gmin)*(255/gdelta));
        // compute the relative frequency histogram h'(n)
        histo[i] = histo[i]/(xd*yd);
}

// compute mapping f(n)
// equal[] is the mapping function f(n)
for (i=0; i<max_grey; i++){
        hsum = hsum + histo[i];
        fequal[i] = hsum*(max_grey-1) + 0.5;
        if (fequal[i] > fmax) fmax = fequal[i];
        equal[i] = unsigned char (fequal[i]);

}
// xd and yd are dimensions of original image
// equalized[][] is the equalized image
// Use of this temporary array is for illustration only, can use pointer, e.g., *indexr
// equal_histo[] is the histogram array of the equalized image
for (j=0;j<yd;j++){
        for (i=0; i<xd; i++){
                equalized[i][j] = equal[array[i][j]];
                equal_histo[equalized[i][j]] +=1;
        }
}
// scale the equal_histo[] for display purpose
gmin = large_number;
gmax = small_number;
for (i=0; i<max_grey; i++){
```

```
            if (equal_histo[i]< gmin) gmin = equal_histo[i];
            if (equal_histo[i]> gmax) gmax = equal_histo[i];

    }
    // equal_histogram[] is the scaled histogram array
    gdelta = gmax - gmin;
    for (i=0; i <max_grey; i++)
          equal_histogram[i] = unsigned char ((equal_histo[i] - gmin)*(255)/gdelta));

    // combining the scaled histogram arrays to display.
    // histogram_image[][] is the image of the two histograms
    // use of this temporary array is for illustration only, can use pointer, e.g., *indexr
    for (j = 0; j < 512; j++){
      for (i=0; i< 256; i++){
        if (j < 256 ){
            // histogram of original image is displayed on top
            if (histogram[i] < (255-j)) histogram_image[i][j] = 255;
            else histogram_image[i][j] = 0;
        }
        else {
            // histogram of equalized image is displayed at bottom
            if (equal_histogram[i] < (511-j)) histogram_image[i][j] = 255;
            else histogram_image[i][j] = 0;
        }
      }
    }
```

4) Results: Use the following images: (a) "Ziyi_Zhang" (x,y) = 225, 160; (b) "arteries", (x,y) = (185,64); (c) "flowers", (x,y) = (270,260); (d) "pattern2", (x,y) = (470, 280) where NxN = 128. Use any additional images and/or values of NxN as appropriate.

B. Non-programming project:

1) Research: Perform research to locate papers and articles related to image binarization using histogram. Select the best three papers. Write a summary of each paper. Compare and contrast the ideas presented in the papers. Attach a copy of each paper to the report.

2) Algorithm development: Develop an algorithm to perform binarization automatically and adaptively using histogram method. Discuss the basic ideas. Provide pseudocode to implement the algorithm. Discuss the expected results and/or problems.

☐ **Project Report:** Follow the required format. Attach these sheets as cover sheets for the report. For programming project, attach printouts of at least the following images: "Ziyi_Zhang_grey.bmp", "Ziyi_Zhang_grey2.bmp", "Ziyi_Zhang_gray16.bmp", "Ziyi_Zhang_spatial512.bmp", "Ziyi_Zhang_equalhisto.bmp", "Ziyi_Zhang_histograms.bmp", "arteries_grey16.bmp", "pattern2_grey.bmp", "pattern2_equalhisto.bmp", "pattern2_histograms.bmp", "arteries_spatial512.bmp", "flowers_spatial16.bmp". Discuss the results and specific implementations Compare the histogram equalized images and the original images. Discuss. For non-programming project, provide pseudocode detailing the algorithm and discuss expected results.

END OF DOCUMENT

Shannon Foss

This project consisted of image formatting and analysis on a bitmap image: bitmap extraction and creation, grey levels, cropping and resizing, equalizations and histograms.

A bitmap consists of a 54 byte header that holds such information as file size, image height and width, etc. and the image data. The image data is made up of red, green and blue values ranging from 0 to 255 that represent each pixel within the picture. After extracting the image data to an array, you may manipulate or analyze the data to form new images or produce information about the image.

Grey Level images are made up of pixels that have values of grey that range between 0 and 255. This is accomplished by averaging the red, green, blue values of each pixel of the original picture and then equalizing them. To create lower grey levels (32 or 64 rather than the normal 256), the values must be stepped: a range of values is now represented by one value. For example, if you wish to display a picture with 32 grey levels, than values 0 through 7 are represented by 0, 8 through 15 are represent by 8, 16 through 23 are represented by 16, and so on until you reach 255. These grey level images will give you a good black and white image of your original color image. However, as you can see from the figures below, the lower the grey level, the less detail there will be in the resulting image.

Fig. 1. flowers_grey.bmp – full grey scale image with all 256 values of grey.

Fig. 2. flowers_grey8.bmp –grey scale image with only 8 grey levels.

Creating a sub-image consists of picking a location within the original image, cutting out a square portion of a specified dimension around it and then setting it up for display. The data pulled from the image is stored in an array, and the header needs to be created for the smaller file.

Fig. 3 flowers_spatial256.bmp – A 256x256 image centered on location (270, 260) from flowers_grey.bmp

From here it is also possible to increase or decrease the display size of the image, to increase the size; a simple repetition of rows and columns in multiples of the increasing factor is needed. To decrease; just remove rows and columns based on multiples of the decreasing factor.

(a) (b) (c) (d)

Fig. 4. – flowers_spatial256.bmp, flowers_spatial128.bmp, flowers_spatial32.bmp, flowers_spatial16.bmp – Here, the captured image: (b) flowers_spatial128.bmp has been enlarged to (a) size 256x256, it has also been reduced to (c) size 32x32, and (d) 16x16.

Equalizations are applied to a picture to even out the tones by changing the frequency envelope of the image. If there is a high frequency of one particular value within the image, then equalizing it will lessen that value and compensate by increasing other nearby values. The sum of the frequencies will stay the same but there will be less peaking frequencies.

As you can see in comparing figure 1 to figure 5, many of the high frequency dark tones and high frequency light tones have been minimized, where as many of the middle tones have been increased. Overall, this gives the picture a washed out look. In some parts of the photo it seems that you lose the detail of its depth, the image seems flat because there are fewer differences between the lighter foreground and the darker background. However, the equalizing lightens the dark areas to expose more detail in places where it was previously too dark to see.

Fig. 5. – flower_equalhisto.bmp – An example of an equalized image (compare with figure 1).

Histograms let you view a representation of the values of the frequencies of the image. The top histogram in figure 6 shows the frequencies of pixel values that are in the image from figure 1. It shows a few large spikes in the lower values, this implies that there is a good deal of very dark values within the image, most likely the water in the picture. There are also two smaller hills within the histogram, one representing the middle greys, perhaps the lillypads, and the other shows the number of lighter values, probably the flowers.

The histogram on the bottom of figure 6 represents the equalized flower image from figure 5. As you can see, the darker values have been spread out, and there are now many more light values present.

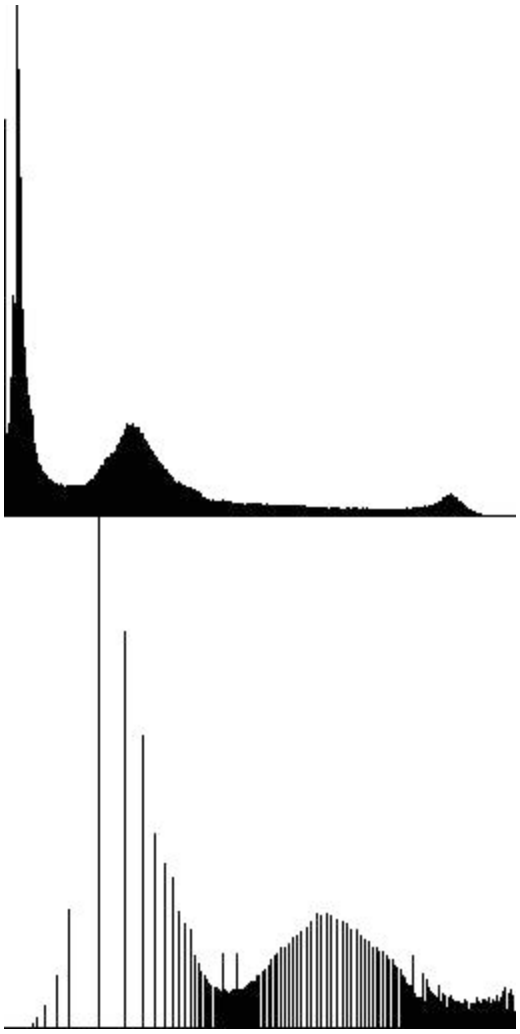


Fig. 6 – flowers_histograms.bmp – Here the histograms of flowers.bmp (top) and flowers_equalhisto.bmp (bottom) are shown. i.e. before and after equalization.

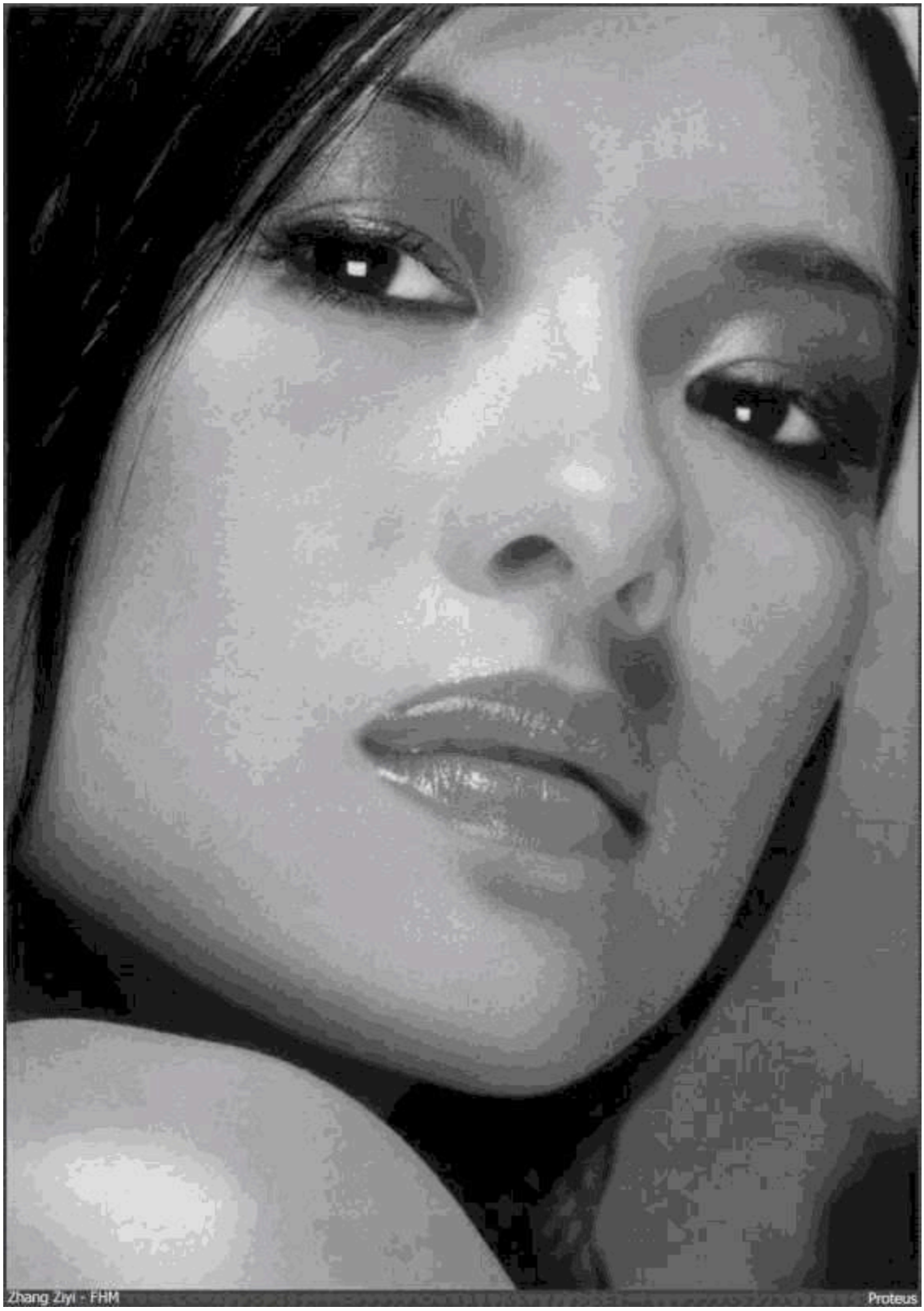
The formatting of a bitmap image is not as difficult as one might think, once you define what needs to be in the header and what data in the image needs to be changed it becomes an easy task. This project was a good task to practice on how bitmaps are created, manipulated and analyzed.

Extra Images

Ziyi_Zhang_grey.bmp

Ziyi_Zhang_grey2.bmp

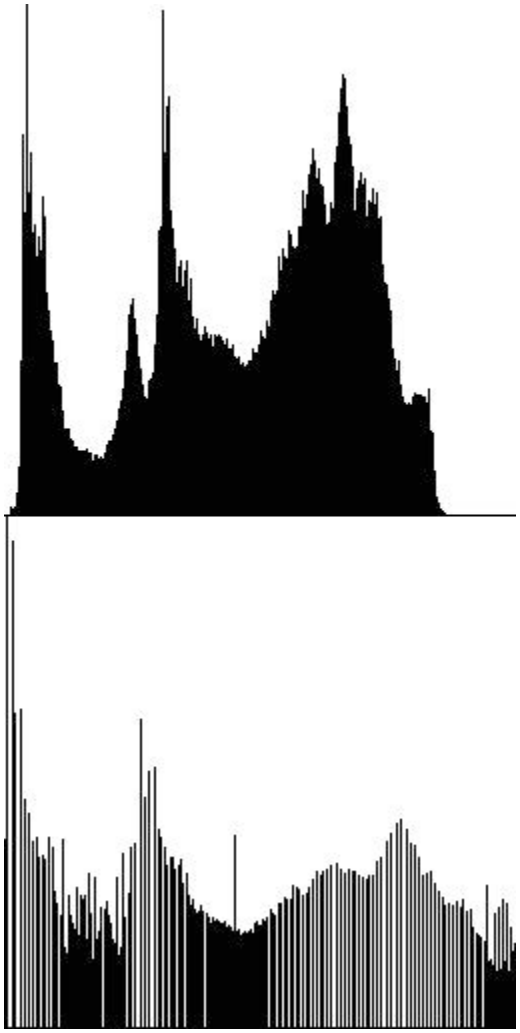
Ziyi_Zhang_gray16.bmp



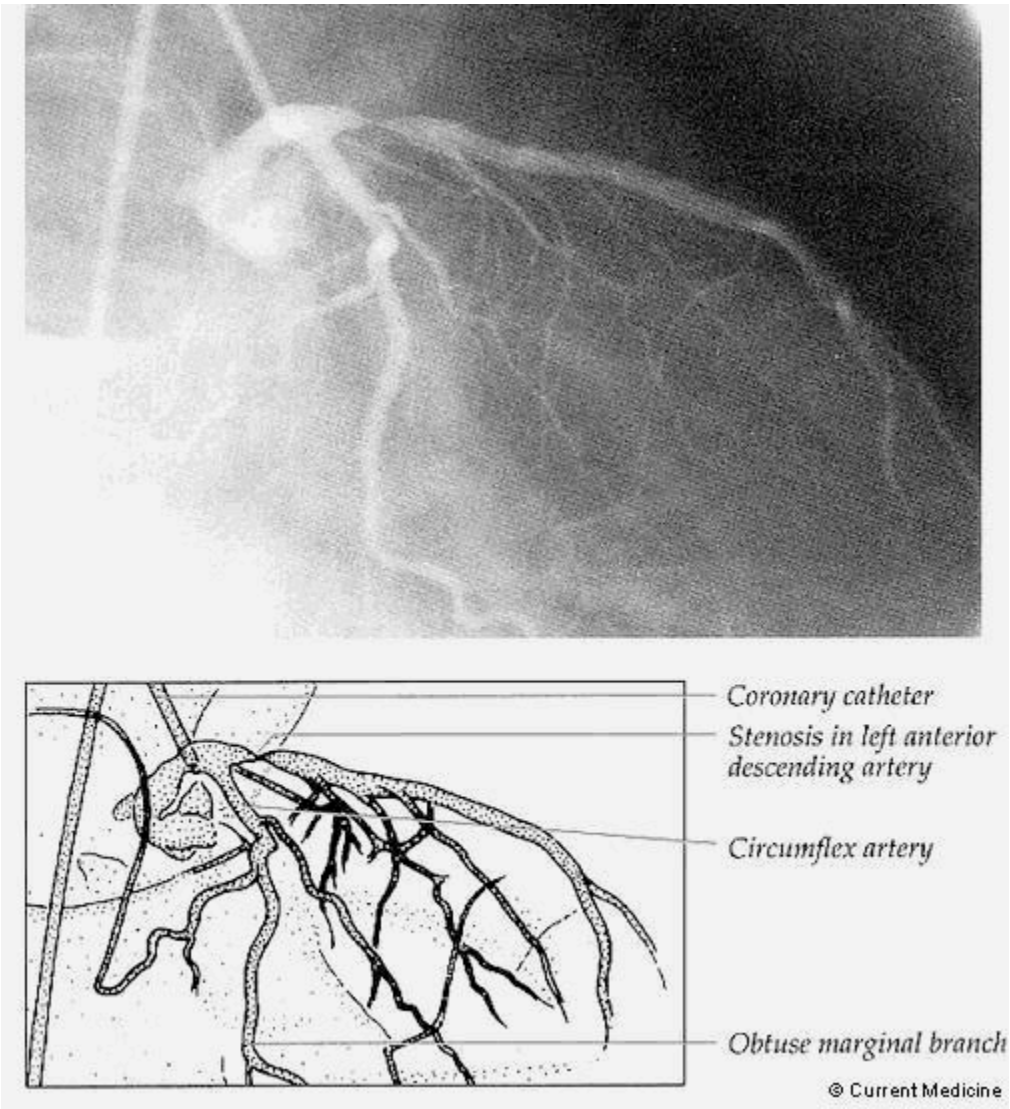
Ziyi_Zhang_equalhisto.bmp

Ziyi_Zhang_spatial512.bmp

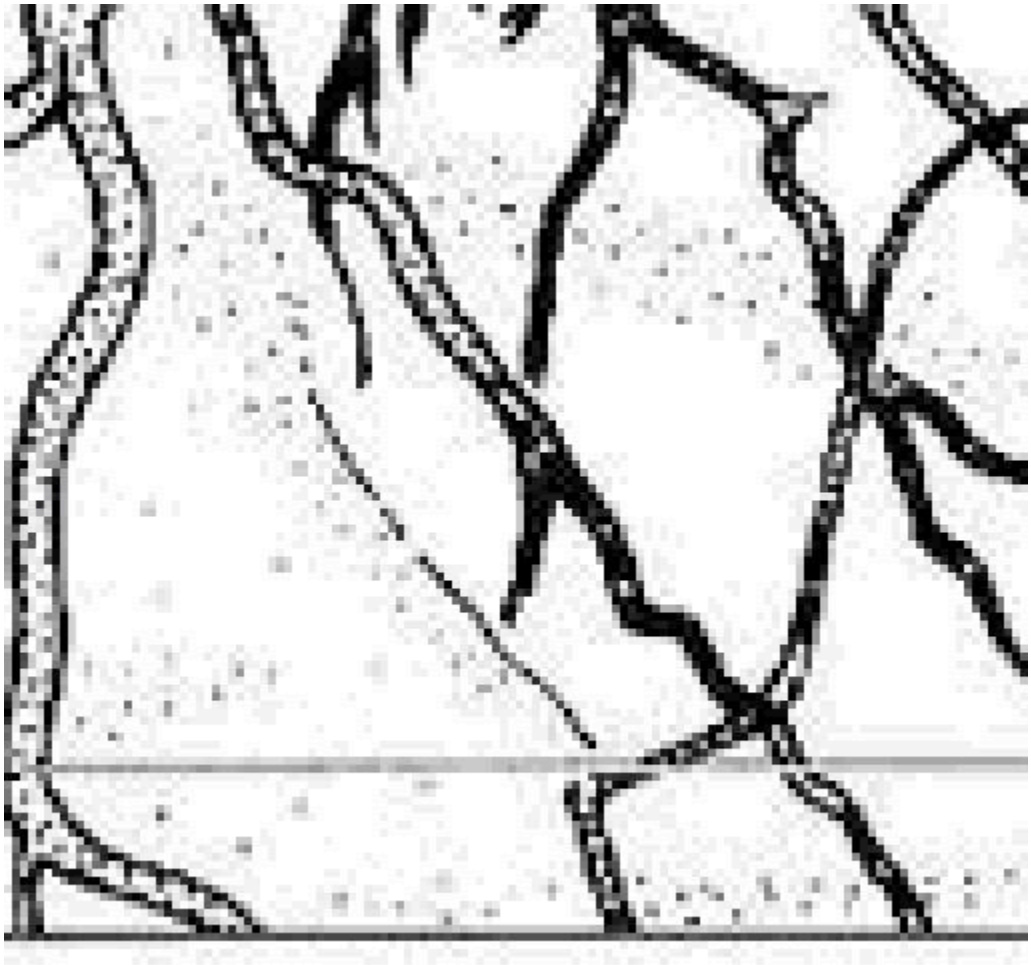
Ziyi_Zhang_histograms.bmp



arteries_grey16.bmp

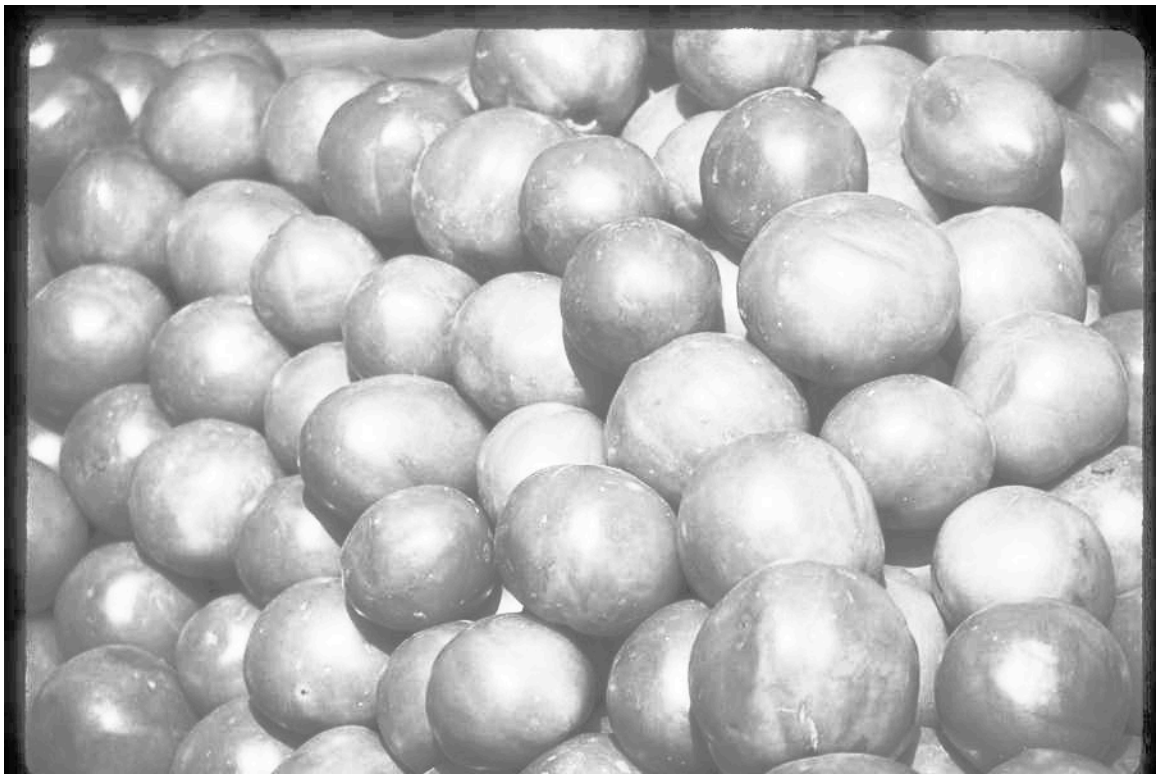


arteries_spatial512.bmp



pattern2_grey.bmp

pattern2_equalhisto.bmp



pattern2_histograms.bmp