Computer Aided Diagnosis

Assignment 2 - Image Processing Basics

Diagnostic Image Analysis Group

February 12, 2016

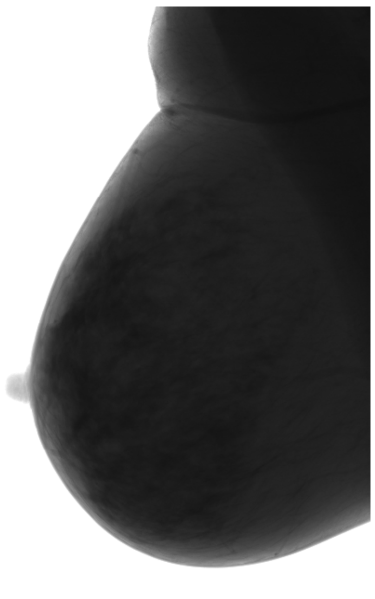
Please send the cpp code, mevislab networks, answers to questions and some screenshots of your results to:

`katharina.holland@radboudumc.nl`
`mehmet.dalmis@radboudumc.nl`

Exercise 1 and 2 are obligatory, exercise 3 is optional.

1 Gray-scale transformation for breast image processing

Description: In this subsection you are asked to create an image processing network to transform a raw breast image (r0100009002mr.dcm) in a way that the information of different tissues inside the breast is significantly more distinguishable. Figure 1(a), shows a sample raw image. In Figure 1(b) you can see the corresponding processed image which seems to be much more informative to do the computer aided diagnosis tasks. You can display this image your self by loading p0100009002mr.dcm. Note that the processed image you


(a) Raw Image

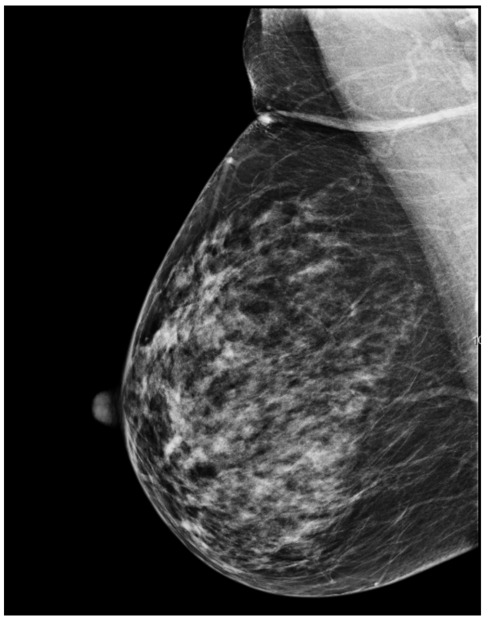

(b) Processed Image

Figure 1: Raw and processed breast images

see here is the result of a relatively complicated task. Each manufacturer has

its own way of processing. In this assignment you will try to do some simple processing steps to enhance the different tissue structures. We do not expect you to derive such a perfect result. In this exercise we will use Look Up Tables (LUT) to represent the transformation function.
Please follow the following steps and answer the specified questions:

1. Create a very simple network using the "LocalImage" and "View2D" modules so that you can see and compare raw and processed images.

2. Now we need a Logarithmic transformation. You can do this with an "Arithmetic1" module.

3. Observe the approximate range of intensity values you see in the breast area, by moving the mouse around the areas with the most visible tissue structures. Notice that this is the range of intensities we want to have stretched in the processed image, so that we have an enhancement of this area.

4. Confirm this range, by creating a histogram of the pixel intensities. Do the following steps:

   (a) Put an instance of the "Histogram" module in your network. Right click on this module and select *Show Example Network*.

   (b) Now you can see a sample network for histogram demonstration. Select the whole network except for the "LocalImage" module and paste it into your network.

   (c) Connect the output of the Logarithmic transformation to the left input of the "Histogram" module and double click on "SoRender-Area" to see the histogram.

   (d) Double click on the "Histogram" module. Uncheck *Use bin size one representation* and use an appropriate *Bin Size* like 0.1.

   (e) Was the Logarithmic transformation a useful step? What is the problem? Solve it with the module "ImagePropertyConvert".

   (f) (Question 1) There you can see a high peak around 9. What is that peak?

   (g) (Question 2) Explore the different options of "SoDiagram2D" until you see a histogram similar to Figure 2. What options should be changed?

   (h) (Question 3) What do these two peaks stand for? (Hint: Use the "IntervalThreshold" module to support your guess.)

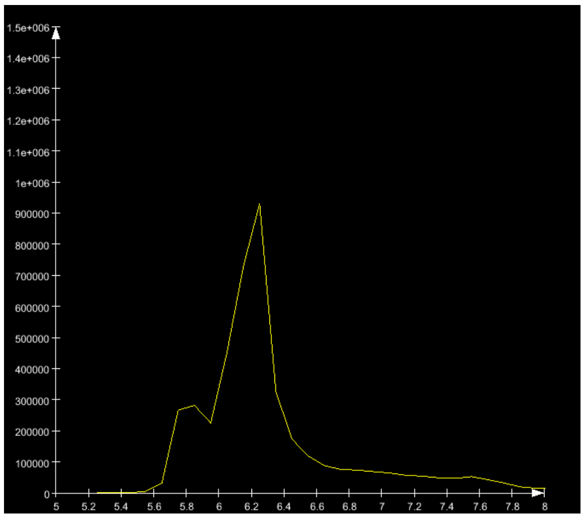


Figure 2: Histogram of raw breast image.

5. In order to be able to use and manipulate LUTs create a network like what you see in Figure 3.

6. Double click on "SoRenderArea" so that the shape of the LUT is demonstrated.

7. Double click on the "LUTCurve" module and check the *Rescale* item. You can also place the mouse on *Rescale* and *Rescale Width* items and observe the documentation about the parameters to get an idea what they are.

8. Double click on "LUTPrimitive". Using the menu you see, you can change the properties of your LUT curve. Uncheck *Relative* checkbox. Manipulate the center and width values and see how the function form changes.

9. (Question 4) What *Shape* type (Ramp, Wall, Gauss, Sigmoid) do you think will best fit for the transformation task in hand? Why?

10. (Question 5) According to the range of intensities you got from steps 2 and 3, what values do you suggest for the center and width values?

11. In order to apply your LUT as a gray level transformation, put an "ApplyLUT" module into your network and feed it with your image and the output of "LUTPrimitive" as its inputs. Go with your mouse to the input triangle of "ApplyLUT". What type of input is expected?

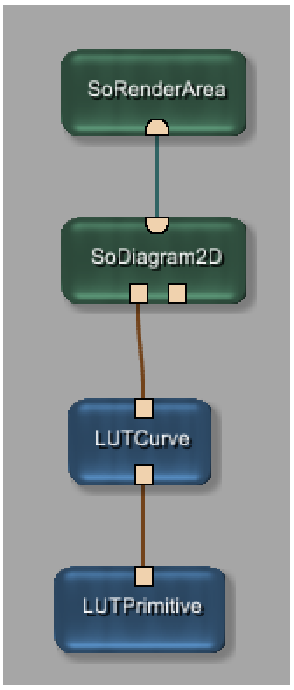


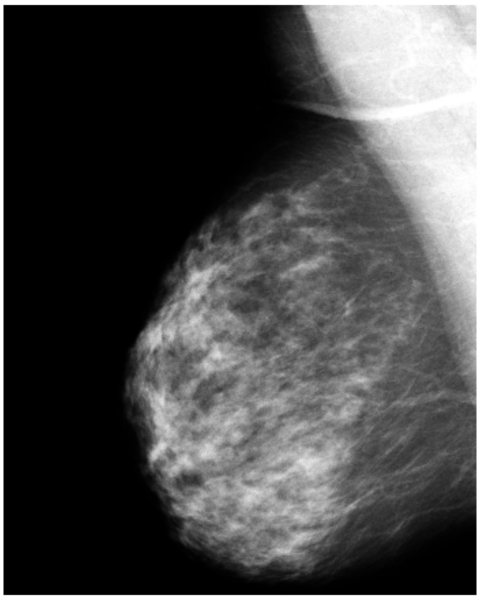
Figure 3: A sample network for creating a LUT

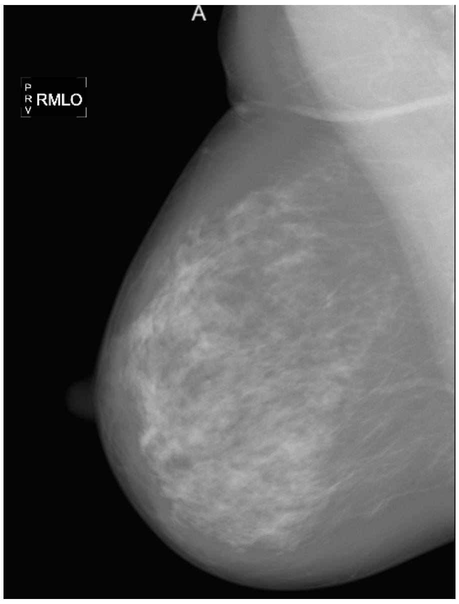
Solve that problem, you might want to use again the ”Arithmetic1” module.

12. If you look at the output of ”ApplyLUT”, comparing it to the processed image, it seems like intensities in dark regions are light and vice versa. So what we need in this stage is an invert transformation: plug in an ”Arithmetic1” module, double click on it and choose invert as its *Function*.

13. (Question 6) Now open the new output image. If everything has gone well, you might see something similar to the image in Figure 4(a). As you can see, there are some parts being missed in the transformed breast image. Why are they lost in this transformation?

14. (Question 7) The method used so far, is semi automatic because you manually provided some of the parameters. Explain how we can fully automate this process?


(a) A probable output for the previous steps.



(b) A sample breast image resulted from applying two transformations

Figure 4: Possible results

2 Histogram equalization

To be able to implement the histogram equalization, we need to create a histogram of the pixel values, make a cumulative histogram, transform the cumulative histogram and finally update the output image. The following steps will guide you through the process:

1. Create a new empty ML module using the project wizard.
2. (Question 8) Calculate the minimum and maximum pixel values of the input image and store them in two variables. Why do you need these?
3. Create an empty histogram with the number of bins equal to the range of values in the input image. You can use a C++ vector at this step. An example for usage of a C++ vector is given below:

```
std::vector<int> myvector (length, initial_value);
myvector[i] = some_value;
```

   The first line creates a vector of integers `myvector`, having `length` number of elements, each initialized with `initial_value`. In the second line, the element with index `i` is accessed and filled with `some_value`. Note that the index of the first element of a vector is `0`.

4. Now fill the histogram with the input pixel values.

5. (Question 9) After the histogram is filled, we need to make a cumulative histogram. In this cumulative histogram, the value of the next bin is the sum of all previous bins. Create the cumulative histogram. What does the value of the last bin represent?

6. (Question 10) Calculate the histogram transformation and scale it appropriately by transforming the cumulative histogram as given by the following equation:

$$h_i = (\frac{L}{NM}) * ch_i \tag{1}$$

   with h_i the ith item of the transformed histogram, ch_i the ith value of the cumulative histogram, NM the number of pixels in the input image and L the range of different gray-levels. (In C++ you have to convert a variable of type integer to a float before using it for a division. You can cast a variable to float by using float(variable).) How do you calculate L?

7. Finally, rescan the image and set the output image with gray-levels g_q, setting $g_q = h[g_p]$ with g_p the input image value.

8. (Question 11) When you use your new ML module, put a **PageToImageExtent** module in front of your ML module. This ensures correct memory handling in your module. Inspect your result by checking the histogram of the equalized image. (In MeVisLab, use a **Histogram, SoDiagram2d** and a **SoRenderArea** module. What does histogram equalization do?

3 EXTRA: Image Registration

Description: Successfully aligning images from a single modality or from multiple modalities enables subsequent processing, analysis, and/or visualization that can aid in the screening, diagnosis, prognosis, treatment, and monitoring of diseases. In the past two decades, a vast amount of research has been performed to develop various models and computational techniques for image registration across a wide spectrum of applications. In this subsection, we are going to explore some abilities of this family of algorithms. Please follow these steps:

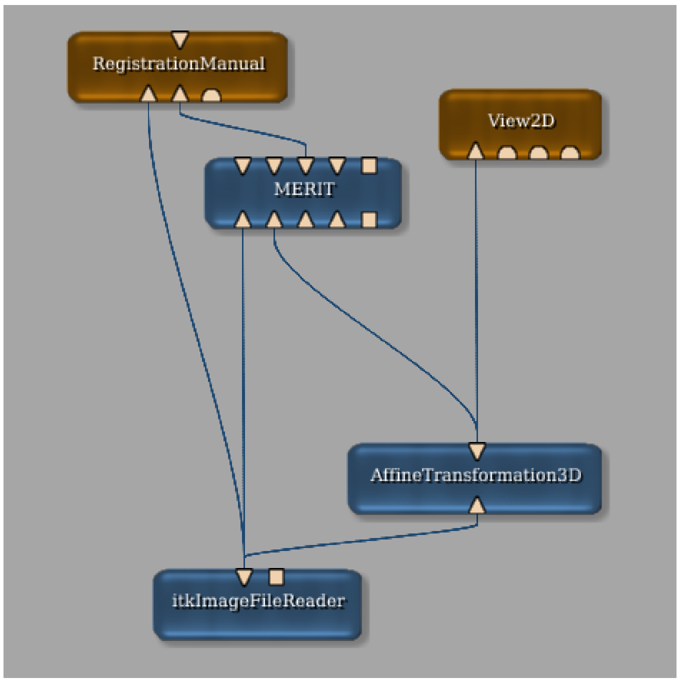


Figure 5: A simple network for applying registration.

1. Open your input image (*_t1_biasCorrected_fl.nii.gz*) with "itkImage-FileReader" and feed it into an "AffineTransformation3D" module.

2. In "AffineTransformation3D" choose some values for Translation X, Translation Y, Rotation Z and observe the transformation on the output image.

3. Insert a "MERIT" module which is basically prepared for image registration- and give the original image and the output of "Affine-Transformation3D" as its first two inputs respectively.

4. To visualize the overlaying of the original and the registered images, put a "RegistrationManual" module into your network and connect the original image and the third output of the "MERIT" module to its input slots. Up to this point, your network should look like the network in 5.

5. Open the "RegistrationManual" module panel, and assign two nicely distinguishable colors for the Reference and Overlay colors.

6. Open the "MERIT" panel, choose *Affine* as Transformation type and *Ssd* as Similarity Type. Press the "Automatic Registration" key and

observe the result of the registration.

7. To better measure how similar (or better to say how different) the original and the registered images are, use "Arithmetic2", "Arithmetic1" and "ImageStatistics" modules to create a sub-network that calculates the mean squared difference of these two images.

8. Feel free to choose different parameters for your transformation! Observe how your similarity measure changes.