in raw format, it is very handy to have a histogram that shows where the majority of gray values ρ can be found. Based on this information, one can choose the optimum data type and range for an initial intensity transform. If one chooses to segment the bony content of an x-ray image, an initial guess for an optimum threshold can be taken from the histogram. Finally, it may be necessary to design an intensity transformation in such a manner that two histograms match each other; this process, usually referred to as *histogram equalization*, may be a necessary step prior to registration of two data sets. Furthermore, we will learn that histograms can also be used to design similarity measures for image registration, as in the case of the *mutual information* algorithm, which we will encounter in Chapter 7.

3.4 Dithering and depth

The task of reducing the depth of gray scale images is rather easy; one can either scale the range of data, or use a windowing function to fine tune visible detail. The issue gets more complicated when reducing the depth of color images. This process, usually referred to as dithering, reduces the color space to a smaller amount of possible colors, which can be given as a LUT. A classical algorithm is *Floyd-Steinberg dithering*, but numerous other algorithms exist. The good news is that in medical imaging, we barely ever encounter color images; even the colorful images provided by PET and SPECT are the result of color encoding of count rates from a nuclear decay process. This color coding is achieved by means of a LUT. However, the literature on general image processing gives numerous details about different dithering methods.

3.5 Practical lessons

3.5.1 Linear adjustment of image depth range

First, it would be great if we would be able to apply a linear ITF to the images read in Section 2.7.1; for this purpose, we have to determine $\rho_{\max}$ and $\rho_{\min}$, and apply Equation 3.1. Here, we transform the MR-image from the `PIG_MR` from its original gray values to an unsigned 6 bit image for good display using the MATLAB `image` function. The script is named `LinearIntensityTransform_3.m`. First, we have to load the image again:

```
1:> fpointer = fopen('PIG_MR','r');
2:> fseek(fpointer,8446,'bof');
3:> img = zeros(512,512);
4:> img(:) = fread(fpointer,(512*512),'short');
5:> img = transpose(img);
```

Next, the minimum and maximum values of the image are determined, and the whole image has to be transformed according to Equation 3.1. Again, we have to call `max` and `min` in order to get the maximum component of the `img` matrix. The result can be found in Figure 3.8, and it does now look pretty familiar compared to Figure 2.13.

```
6:> rhomax = max(max(img))
7:> rhomin = min(min(img))
8:> newimg=zeros(512,512);
9:> newimg = (img-rhomin)/(rhomax-rhomin)*64;
10:> colormap(gray)
11:> image(newimg)
```

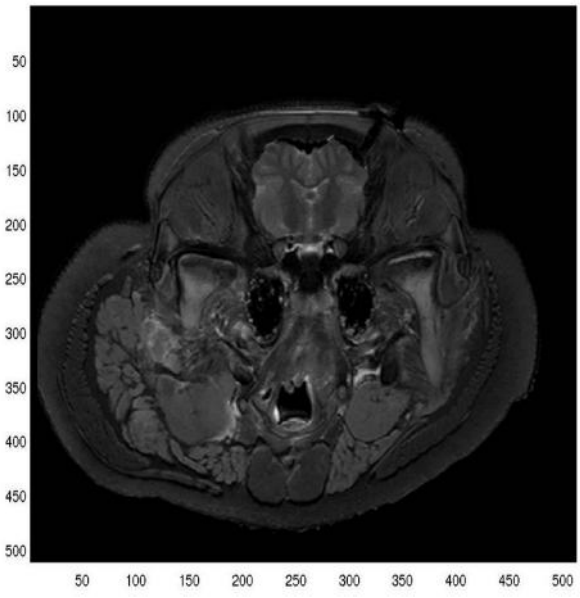

Figure 3.8: The output from `LinearIntensityTransform_3.m`. The MR-image already known from Example 2.7.1 is now displayed using an appropriate gray scale range.

Additional Tasks

- Repeat the same procedure with the data set `PIG_CT`, but transform it to `unsigned 8 bit` and save it as a PGM.

3.5.1.1 A note on good MATLAB programming habits

You may replace the double call to the `max` and `min` function in lines 6 and 7 of `LinearIntensityTransform_3.m` by a single function call:

```
6:> rhomax = max(img(:));
7:> rhomin = min(img(:));
```

The impact on algorithm performance is, however, negligible.

3.5.2 Improving visibility of low-contrast detail – taking the logarithm

In `LogExample_3.m`., we will learn about a very simple way to improve the visibility of low-contrast detail. A simple method to enhance the visibility of low contrast detail is to take the logarithm of the image prior to intensity scaling. Remember that the logarithm is a function $y = \log(x)$ that is zero for $x = 1$, $-\infty$ for $x = 0$, and becomes shallow for growing values of x. Applying the logarithm $\rho' = \log(\rho + 1)$ to a 12 bit DICOM image should therefore result in an image where the dark, low contrast parts of the image appear enhanced. Denote that, as opposed to Equation 3.1, this transformation is not linear.

First, we open a DICOM image from CT showing an axial slice; we transform the range of gray values from $\{\min, \ldots, \max\}$ to $\{1, \ldots, \max - \min + 1\}$ just to avoid the pole of log(0). Next, the logarithm is taken and the result is scaled to 6 bit image depth and displayed. Figure 3.9 shows the result.

```
1:> fp=fopen('SKULLBASE.DCM', 'r');
2:> fseek(fp,1622,'bof');
3:> img=zeros(512,512);
4:> img(:)=fread(fp,(512*512),'short');
5:> img=transpose(img);
6:> fclose(fp);
7:> minint=min(min(img));
8:> img=img-minint+1;
9:> img=log(img);
10:> maxint=max(max(img));
11:> img=img/maxint*64;
12:> colormap(gray)
13:> image(img)
```

Additional Tasks

- Can you think of a similar non-linear transform that emphasizes high-contrast detail? Modify `LogExample_3.m` to achieve this.

3.5.3 Modelling a general nonlinear transfer function – the Sigmoid

In `SigmoidIntensityTransform_3.m`, we demonstrate the impact of the ITF-curves modelled as the sigmoid function $S(\rho)$ from Equation 3.2 on a sample 8 bit CT-slice. First, we have to assign the working volume again, and load the image `ABD_CT.jpg` using `imread`. The JPG-image was saved as gray scale, therefore only one layer is available (as opposed to Example 2.7.3). Finally, the width and height of the read image is determined using `size`. The result is 435 pixels width and 261 pixels height.

```
1:> oimg = imread('ABD_CT.jpg');
2:> size(oimg)
```

In the next step, a vector containing the values for $S(\rho)$ is defined and filled; the parameters `pomega` and `psigma` (= ω and σ in Equation 3.2) have to be defined

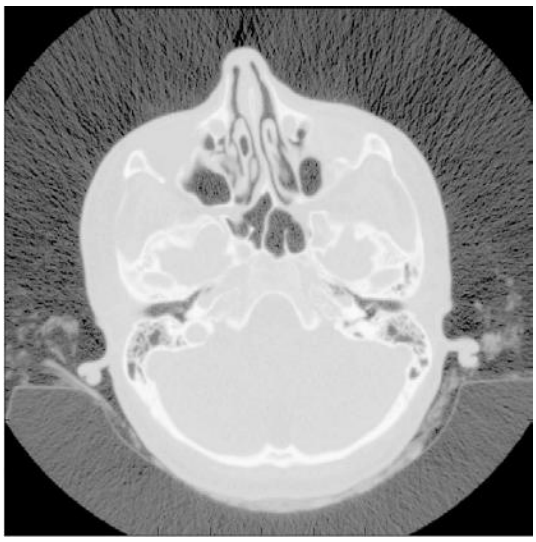

Figure 3.9: Taking the logarithm of an image emphasizes low-contrast detail; in this example, both CT reconstruction artifacts as well as radiolucent structures such as the hair of the patient are emphasized whereas diagnostic detail vanishes. (Image data courtesy of the Department of Radiology, Medical University Vienna.)

as well. Again, the suffix `;` is important in the `for`–loop. Indexing a vector like `sigmoid` starts from 1, but the gray scale values ρ start from zero – therefore the running independent variable ρ has to be augmented by 1 for indexing `sigmoid`.

```
3:> pomega = 127;
4:> psigma = 25;
5:> sigmoid = zeros(256,1);
6:> for rho=0:255
7:> sigmoid(rho+1,1) =
64*1/(1+exp(-((rho-pomega)/psigma)));
8:> end
```

We can now take a look at the curve $S(\rho)$, $\rho \in \{0 \ldots 255\}$ using the `plot`–command, which generates output similar to Figure 3.3. The interesting concept here is that `sigmoid` is indeed a mathematical function represented by numbers that belong to discrete pivot points. In the next steps, the original values of image gray value ρ are transformed using the vector `sigmoid`, which translates these gray values according to the modelled ITF. Denote that we use a little trick here: being a function defined as discrete values, the Sigmoid is saved as pairs $(x,\ f(x))$. However, we only have a simple vector `sigmoid` that holds all the dependent values from Equation 3.2; the independent variable is the index of the vector here. Therefore we can determine the gray value `rho` in the original image and replace it by the the value of the

`sigmoid` vector at position `rho+1` since the gray values range from 0 to 255, but indices in MATLAB are given as 1 ... 256.

```
9:> plot(sigmoid)
10:> transimage = zeros(261,435);
11:> for i=1:261
12:> for j=1:435
13:> rho = oimg(i,j);
14:> transimage(i,j)=sigmoid(rho+1,1);
15:> end
16:> end
```

After calling `colormap(gray)`, `image(transimage)` and `image(oimg)` show the transformed and the original image.

Additional Tasks

- Repeat the procedure for parameter pairs $\omega/\sigma = 180/15$ and $100/45$. Results are shown in the images in Figure 3.10. Try it!
- The basic method in windowing is to cut off the intensity values that are outside the window, and to scale the remaining pixels. Therefore, *windowing* is a simplification of the sigmoid-shaped ITF in a sense that the dynamic range is a given as a linear function. The general shape of a linear function is given as

$$f(x) = a * x + b$$

where a is the gradient of the linear function, and b gives the value for $f(0)$. Implement a windowing operation. Do you see a correspondence of `pomega` and `psigma` and the window center and width?

3.5.4 Histograms and histogram operations

As a first example in `Histogram_3.m`, we will derive a number of histograms of varying bin number from the example CT-slice `ABD_CT.jpg`. We define the working directory, read the image, determine its depth of gray values, and allocate memory for a number of histograms of varying bin width.

```
1:> img = imread('ABD_CT.jpg');
2:> depth = max(max(img))-min(min(img))
3:> hist16 = zeros(16,1);
```

We have found out that our image has a depth of 8 bit, and we have allocated a vector `hist16`, which will be our histogram with 16 bins. The next step is to determine the minimum gray value of the image, which is the starting point for the first bin, and the bin width; in order to cover a range of 256 gray values, we need a bin width of 16 for a histogram with 16 bins according to Equation 3.3. We can now sort the gray values ρ into the histogram bins. The image has a width of 435 pixels and a height of 261 pixels; we design the `for`-loops accordingly:

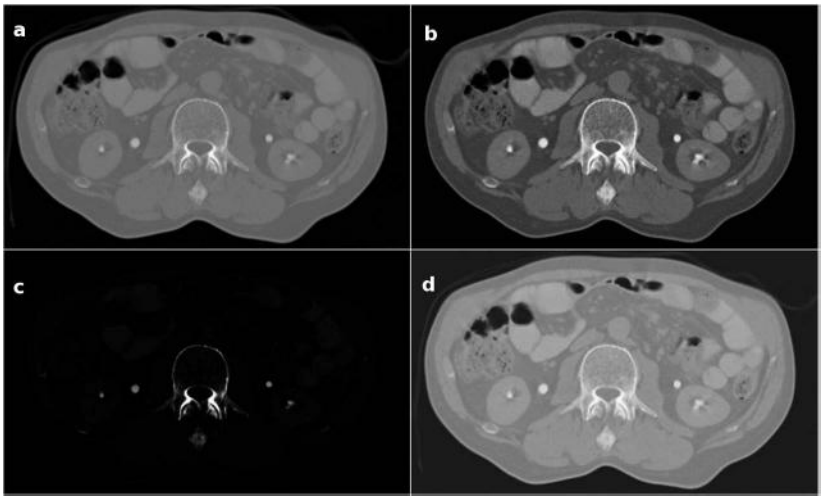

Figure 3.10: The impact of the three sigmoid functions $S(\rho)$ from Figure 3.3 on an 8-bit CT slice of the abdomen (upper left, denoted as **a**). The first function with $\omega = 127$ and $\sigma = 25$ gives a well balanced image where the extreme dark and extreme bright areas are slightly suppressed, and the middle gray values translate more or less linear (upper right, denoted as **b**). A sigmoid function with $\omega = 180$ and $\sigma = 15$ suppressed the darker areas and highlights only the extremely intense image content such as cortical bone or contrast-agent filled vessels (lower left, denoted as **c**). Finally, a wide and shallow sigmoid ($\omega = 100$ and $\sigma = 45$) gives a dull image. Dark areas appear brighter, revealing additional low contrast detail (such as the patients clothing), whereas high-contrast detail is darker and less intense (lower right, denoted as **d**). The sigmoid functions can also be considered a model for detector efficiency – for instance in x-ray imaging – if the original gray values are considered to be an equivalent of incident energy. (Image data courtesy of the Department of Radiation Oncology, Medical University Vienna.)

```
4:> for i = 1:261
5:> for j = 1:435
6:> rho = img(i,j);
7:> b16 = floor(rho/17.0)+1;
8:> hist16(b16,1)=hist16(b16,1)+1;
9:> end
10:> end
```

We have used a nasty little trick for determination of the appropriate bin; the result of the division $\rho/(\text{Bin Width} + 1)$ is rounded to the lower integer by using `floor`. This operation should result in the appropriate bin number, ranging from 0 to the number of bins minus one. Since indices in Octave and MATLAB range from 1 to the maximum number of bins, the bin number has to be increased by 1. In the

next step 1 was added to the resulting bin. For a visualization of the histograms, the following command is helpful:

```
11:> bar(hist16)
```

The output from `Histogram_3.m` can be found in Figure 3.11.

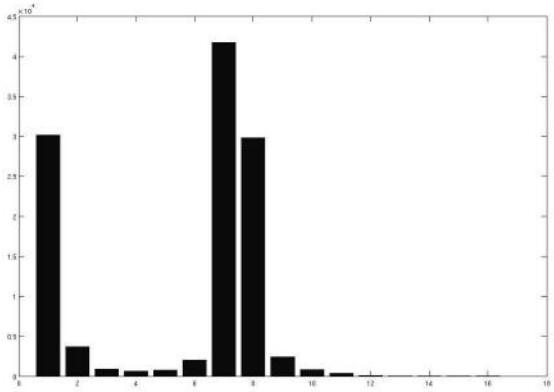

Figure 3.11: The output from `Histogram_3.m`, where a histogram of 16 bins is computed from the 8 bit image `ABD_CT.jpg`.

For visualization of the histogram using a spreadsheet program, one can also save the histogram as a text file, and import it into a standard spreadsheet program; the command to be used is `save`; the visualization of such histograms in a spreadsheet program such as *Microsoft Excel* or *OpenOffice* can be found in Figure 3.7.

```
12:> save('Histogram16.txt','hist16','-ascii');
```

Additional Tasks

- Generate a histogram with 64 and 256 bins.
- Compute a histogram with 64 bins for the 12 bit dataset `PROSTATE_CT`. It is a single DICOM slice of 512×512 pixels, 16 byte depth, and a header of 980 bytes.
- Use the MATLAB function `hist` to draw the histograms.

3.6 Summary and further references

Transformations in intensity space play an extremely important role in medical image processing and image display in general, especially since medical images are usually recorded with an image depth beyond human perception. Transformations can range from applying rather simple analytical functions like taking the logarithm to more sophisticated models such as parametrized sigmoid curves or special windowing operations, which

enhance the details visible within the signal range of a certain tissue type. The distribution of gray values is represented by a histogram.

Literature

- R. C. Gonzalez, R. E. Woods: Digital Image Processing, Prentice-Hall, (2007)
- A. Oppelt (ed.): Imaging Systems for Medical Diagnostics: Fundamentals, Technical Solutions and Applications for Systems Applying Ionizing Radiation, Nuclear Magnetic Resonance and Ultrasound, Wiley VCH, (2006)
- W. Burger, M. J. Burge: Digital Image Processing – An Algorithmic Introduction using Java, Springer, (2008), `http://www.imagingbook.com`