EIKONA

Digital image processing package

Thessaloniki 1997

Manual, Part II

EIKONA for Unix

Users guide

Version 3.2

The JPEG input/output routines were based in part on the work of the Independent JPEG Group.

The GIF, BMP, TGA input/output routines were base in part on the PBMPLUS toolkit.

The scanner interface was based in part on the TWAIN Toolkit, Release 1.5.
The TWAIN toolkit is distributed as is. The developer and the distributors of the TWAIN toolkit expressly disclaim all implied, express or statutory warranties including, without limitation, the implied warranties of merchantability, noninfringement of third party rights and fitness for a particular purpose. Neither the developers nor the distributors will be liable for damages, whether direct, indirect, special, incidental, or consequential as a result of the reproduction, modification, distribution or other use of the TWAIN toolkit.

Distributor :

Contents

1

EIKONA for Unix

1.1 Introduction

EIKONA for Unix is a powerful, but yet simple to use, digital image processing software package that runs under X-Windows and implements over 150 image processing routines in the following areas:

- image display, scanning and printing
- image thresholding, clipping
- addition, subtraction and multiplication of images
- and, or, xor bit-level operations between images
- addition/multiplication of an image by a constant
- various image noise generators
- two dimensional filters including adaptive and nonlinear filters
- histogram and cdf histogram computation and equalization
- matrix histogram and cdf matrix histogram computation
- image enhancement and sharpening
- region segmentation
- edge detection
- morphological filters
- image transforms
- color coordinate transformations
- image mosaicing, registration and watermarking

EIKONA user interface is based completely on pull-down menus and dialog boxes. As a consequence, it is extremely easy to operate even for users not very familiar with image processing. Images are stored on image buffers. All that is needed to apply an image processing function to an image is to specify the source and destination image buffer and the related parameters. Furthermore, implemented functions are grouped into categories according to the type of operation that they

carry out, in order to facilitate the search for a specific task. The user can choose the image region where processing is to be performed. Multiple images can be displayed on the screen at the same time, a valuable feature when comparing the results of different processing functions on the same image.

EIKONA supports the following file formats : TGA, uncompressed monochrome TIFF, uncompressed BMP, JPEG, GIF and binary (raw) images. Conversion from one image file format to another is also possible.

1.2 Hardware Requirements

EIKONA for Unix can run on any Unix machine which supports the X-Windows system, release 4 or higher. Since many image processing applications are rather time consuming and since images, especially color images, require large amounts of memory, a minimum configuration of at least 16 MBytes of RAM is recommended. Also, in order to take advantage of the image display capabilities of EIKONA for Unix, a color monitor with a 24-bit (True Color) video card capable of displaying 16 milion colors is required.

1.3 Overview

EIKONA maintains a number of buffers on the computer RAM for the storage of images. Grayscale images are internally stored as two-dimensional arrays of unsigned characters that, from now on, will be referred to as *BW* (Black & White) image buffers. The form of the BW image buffers is described in detail in [PIT93]. These buffers are numbered sequentially starting from zero.

Color images are stored on *color* image buffers. In EIKONA for Unix, color image buffers are separate entities from BW image buffers. However, each RGB color buffer consists of three BW buffers. Red component occupies the first buffer, green the second and blue the third one. A special function of the package permits the splitting of a color buffer to its R, G and B components. Color image buffers are numbered sequentially starting from zero. However, two buffers, either BW or color, cannot share the same number.

EIKONA for Unix does not initially allocate any image buffers. They are allocated dynamically when an image is loaded from a file or when the result of an image processing function is stored in a new buffer.

Some functions implemented in EIKONA (e.g. image transforms, color coordinate system transformations), produce as an output two-dimensional float matrices. EIKONA allocates two-dimensional float buffers dynamically when the result of an image processing function has to be stored in a new float buffer. EIKONA

also allocates 2 one-dimensional float buffers (referred to as vectors) that are used internally in some operations e.g. histogram calculation. The contents of these vectors can be saved to disk in decimal format in order to be examined.

Any image processing algorithm implemented in EIKONA can be applied to an image stored in some buffer by simply selecting the corresponding option from the menu. The related dialog box is then presented on the screen, prompting the user to fill in the necessary parameters. Almost all operations require from the user to specify the input (source) buffer (or buffers) and the output (destination) buffer. Selecting a non-existing (not allocated) buffer number as output buffer causes the creation of a new buffer. The given buffer number is assigned to it. Selecting the input buffer to serve also as the output buffer is not prohibited but since many image processing algorithms cannot be executed "in-place", this is not recommended (unless the user knows that the processing routine is a pixel-wise one). Selecting an existing (allocated) buffer number as output buffer causes one of the following: If the type (BW/color) and size of the selected buffer match that of the given image processing algorithm output, the contents of the buffer are destroyed and replaced by the results of the output. If the type (BW/color) or size of the selected buffer does not match that of the given image processing algorithm, the program does not perform the given operation and displays an error message.

EIKONA can open image files and store their contents on image buffers. It supports the file formats binary (raw), TGA, BMP, GIF, TIFF and JPEG. Processed images that are stored on image buffers can be saved on disk files as well.

The contents of a black & white or color image buffer can be displayed by clicking the "Display" command in Black & White or Color menus respectively. When the user chooses to display an image buffer whose one or both dimensions exceed the screen dimensions, a scrollbar appears on the display window to allow viewing the whole of the image.

The coordinates of an image pixel and its value can be displayed on screen, at a separate modeless window. EIKONA allows multiple images to be displayed at the same time. Each image buffer is displayed in a different window. Changing the contents of the buffer automatically changes the image displayed in the associated window.

By default any image processing function is applied to the entire image buffer. However, the user can select a region of interest (ROI), i.e. a part of the image to be processed, by clicking the middle mouse button over the upper left corner of the region of interest and keep it pressed while moving to the lower right corner. During the selection operation, a rectangle shows the selected area. The user can undo the region selection and select the whole image by pressing the right mouse button while the cursor is on an image window. A beeping sound verifies the ROI resetting.

Various filtering operations require a float or integer coefficient mask. These masks, which are read by EIKONA without user intervention, are contained in files float.par and uc.par. The user must edit these files prior to applying the filtering operation and fill in the filter coefficients. The contents of the float.par file could

be the following:

```
3
3
0.11111
0.11111
0.11111
0.11111
0.11111
0.11111
0.11111
0.11111
0.11111
```

The first two numbers show the column and row number of the floating point mask (3×3 in this case). Nine floating point numbers follow that correspond to the mask elements written in a row-wise manner. This mask, when used with convolution corresponds to the moving average filter. The contents of the uc.par file could be the following:

```
3
3
0
1
0
1
1
1
0
1
0
```

The first two numbers show the column and row number of the integer mask (3×3 in this case). Nine integer numbers follow that correspond to the mask elements written in a row-wise manner.

General initialization information, such as block and search region sizes for manual mosaicking routine and initial values for color transformation routines, is contained in the text file eikona.par. The user can modify this file to user-define these values. Typical contents of this file are the following:

```
64
64
1234
11
9
0.2
```

```
0.1
0.3
```

The first two numbers represent the width and height of the thumbnail JPEG image, respectively. The third number represents the default seed for watermarking routines. The fourth and fifth numbers represent the block and search region sizes for manual mosaiking routine. Finally, the last three numbers represent the default color reference values X_0, Y_0, Z_0, respectively, used in color transformation routines.

1.4 Using EIKONA for Unix

To run EIKONA for Unix type "eikona" at a Unix shell.

The EIKONA main window appears on your screen. By default, the menu and text font is the standard font of the system on which EIKONA runs. You can change the default font using the following procedure:

- Create a text file called .Xdefaults in your home directory, or edit the existing .Xdefaults file
- Add the folowing line: eikona*fontList: (font-name)
  where (font-name) is the font encoding as specified by Xlib. A typical line would be the following:
  $eikona * fontList : -adobe - helvetica - medium - r - normal - -14 - 100 - 100 - 100 - p - 76 - iso8859 - 1$
  You can see all the available fonts in your system by typing "xfontsel" at a Unix shell.
- Save your .Xdefaults file and run "xrdb -merge .Xdefaults" to install the specified font.

The new font is permanently associated with EIKONA. If you want to remove it, you can use the procedure described above to change the default font, or you can erase the line concerning EIKONA if you want to use the system font again.

The menu that appears on the top side of the window has five Menu items i.e. **File, Black and White, Color, Arts, Help**. By clicking each of them a pull-down menu appears that contains several options. **File** menu contains file input/output operations as well as other general tasks like file printing, buffer manipulation etc. **Black and White** menu contains all the image processing functions operating on Grayscale images. **Color** menu lists all color image functions provided by EIKONA. **Arts** menu contains image mosaicing, registration and watermarking functions. Finally **Help** provides help on various image processing routines. In the following each of the four menus will be presented in more detail. It must be noted that EIKONA routines have extensive error handling capabilities. If an error is detected a relevant message is displayed and the execution stops.

1.4.1 File

New: Create a new buffer. The user is asked to supply the number of the buffer to be created, its size and its type (i.e. BW or Color). This operation is useful for creating large buffers and storing smaller images to them.

Open: Load an image file from disk. Supported image formats include TGA (Color and Black & White, certain types), TIFF (monochrome, uncompressed), BMP (uncompressed), JPEG and GIF. The user is asked to supply the image file name, the image format and the number of the buffer where the image is to be stored.

Save: Save an image buffer to disk. Supported image formats include Binary (raw), TGA (Color and Black & White, certain types), TIFF (monochrome, uncompressed), BMP (uncompressed), JPEG and GIF. Binary color images are saved in three different files, one for each color component. The three files have extensions ".r", ".g" and ".b". Thumbnail images can also be saved in JPEG format.

Save Matrix: Save an matrix buffer to disk. The user is asked to supply the buffer number to be saved and the file name. Only BW buffer are supported.

Import Raw: Load a Binary (raw) image file from disk. Raw images must be stored row-wise, the first row being the top row. The user is asked to supply the type (BW/color), the number of the buffer where the image is to be stored as well as image dimensions. For BW images the user supplies the full file name of the image. Binary (raw) color images consist of three different files, one for each color component. For these images user either supplies the file name of the image without extension, or selects one of the three files from the file list. The three files must have extensions ".r", ".g" and ".b".

Display coordinates: This is a toggle (on/off) option. When it is on, and the cursor is over an image display window, the cursor coordinates are displayed at a separate modeless window. The upper left corner of the image is the coordinate system origin. For black & white images the grayscale value of the current pixel is also displayed, while for color images, the component pixel values are presented, according to the current color coordinate system. The user can select a certain color space by pressing the right mouse button while the cursor is on the coordinate window, and selecting a color space from the displayed list.

Print: Save a binary image to a file that can be printed to a HP Laserjet printer.

Dump Image: Write the contents of Region of Interest of a BW image buffer to a file in decimal format (ASCII).

Dump Matrix: Write the contents of Region of Interest of a BW matrix buffer to a file in decimal format (ASCII).

Dump Vector: Write the contents of a one-dimensional buffer (vector) to a file in decimal format (ASCII).

Dump Matrix Histogram: Write the histogram of a matrix buffer (X and Y values) to a file in decimal format (ASCII).

Buffer status: Display and destroy the available image buffers. A list of the available image buffers with their type and size is displayed in a window. The user can select by the mouse one or more (CTRL+Left Mouse Button) buffers to destroy. However, the user has to destroy manually the image windows which display images of the destroyed buffers.

Matrix status: Display and destroy the available matrix buffers. A list of the available matrix buffers with their type and size is displayed in a window. The user can select by the mouse one or more (CTRL+Left Mouse Button) buffers to destroy.

Exit: Exit EIKONA.

1.4.2 Black and White

This menu contains all routines for the processing, analysis and display of black and white images. In most cases, the processing routines have one source BW image buffer, one destination BW image buffer and a number of parameters. Processing is performed only within the region of interest (ROI). ROI definition is described in the Overview section of this chapter. It must be noted that the program automatically displays the result of an image processing operation in a new window. If the destination buffer is already displayed in a window on the screen, the user has to destroy the old window.

This part of the manual does not contain detailed description of the various routines, since they are described in detail in the manual of the EIKONA library.
Basic: Sub-menu that contains a number of basic image operations. These include clearing the contents of an BW buffer, copying the contents of a buffer to another buffer, bit-level operations between images (and, or, xor), adding a constant to an image, multiplying an image with a constant, adding/subtracting two images, image halftoning etc.

Processing: Sub-menu that includes negation of an image, image sharpening, image zoom/decimation and also additive and multiplicative noise generators (Gaussian, Uniform, Laplacian) that can be used to corrupt an image for simulation purposes.

Analysis: Edge detection algorithms (Sobel, Laplace, etc.), line and point detection functions, various algorithms for edge following and related algorithms like Direct and Inverse Hough transform. Region segmentation algorithms including thresholding and counting. Texture description and shape analysis algorithms, along with thinning and pyramid methods. Finally, Image Histogram, and Cdf Histogram are also included in this menu.

Transforms: this sub-menu includes the well-known Fast Fourier Transform (FFT) for one and two dimensions and the Discrete Cosine Transform. Also Power Spectrum Density estimation, image convolution etc.

Filtering: Sub-menu containing some of the most widely used filters in the Digital Image Processing. These include Histogram equalization, moving average, median, minimum, rank filter, L-filter and many more.

Nonlinear filtering: Some more nonlinear operators like morphological operators for grayscale and binary images (opening-closing, erosion-dilation), Signal Adaptive Median filter, hybrid, multistage, weighted, separable, median filters, homomorphic filter, harmonic filter etc.

Display: Display grayscale images. The user specifies the buffer to be displayed. The display window shows the number of the displayed buffer. It can be moved around and it can be closed as any other window.

1.4.3 Color

This menu contains all routines for the processing, analysis and display of color images. In most cases, the processing routines have one color image buffer, one destination color image buffer and a number of parameters. Processing is performed only within the region of interest (ROI). ROI definition is described in the Overview section of this chapter. It must be noted that the program automatically displays the result of an image processing operation in a new window. If the destination buffer is already displayed in a window on the screen, the user has to destroy the old window.

This part of the manual does not contain a detailed description of the various routines, since they are described in detail in the manual of the EIKONA library.
Basic: Sub-menu containing basic color image operations like bit-level operations (and, or etc.), buffer copy and clear operations, addition and multiplication of the three RGB components of an image with the same constant etc.

Processing: Noise generators for the three components of a color image, color image filters like marginal median and marginal minimum and also erosion-dilation operators are included in this sub-menu.

Analysis: Color image edge, line and point detectors.

Color Representation: This sub-menu contains a number of color representation system transformations. Many of the color coordinate systems that are included, require float color coordinates. Therefore float buffers must be allocated for such transforms.

Display: Display color images. The user specifies the buffer to be displayed. The display window shows the number of the displayed buffer. It can be moved around and it can be closed as any other window.

Split Channels: Split a color image to its components. The user is asked to supply the source color image buffer and the three destination BW image buffers. The three channels of the color image are copied to the BW image buffers.

Split Matrices: Split a color matrix to its components. The user is asked to supply the source matrix buffer and the three destination BW matrix buffers. The three channels of the color matrix are copied to the BW matrix buffers.

Combine Channels: Combine three BW image buffers to a color image buffer. The user is asked to supply the three source BW image buffers and the destination color image buffer. Each of the three BW image buffers is copied to the corresponding channel of the color image buffer.

1.4.4 Arts

Mosaic : Perform mosaicing of a set of images.

Registration : Perform registration of two images by selecting a set of feature points.

Watermarking : Cast or Remove a digital watermark on an image.

1.4.5 Help

This version of EIKONA does not support On-line Help.

2

Users guide

2.1 How to load, display and save a binary (raw) grayscale image

This section will describe how to load and display grayscale as well as color images. We assume that we are already in the environment of EIKONA for Unix. We also assume that the current graphics display configuration can display at least 256 colors. However, the display mode of 16 million colors that is currently available on most super VGA cards is highly recommended. Check your X-Windows setup to ensure that you have installed the proper graphics mode.

Let us suppose that you have an image of dimensions 256×256 named BAB256.DAT in binary (raw) format on your current directory. Binary (raw) images are stored in a row-wise manner. Each pixel is stored in one byte. The first byte in the file corresponds to the upper left corner of the image. BAB256.DAT can be loaded as follows:

- Choose option **File** by mouse.
- Choose option **Import Raw**.
- A dialog box appears on screen. Click the mouse on the file name BAB256.DAT.
- Give image number 0 in order to load this image file to the image buffer 0. Fill the image width and image height text boxes the value 256. Choose OK. The image is now loaded on image buffer 0 and it is automatically displayed on screen.

The displayed image is shown in Figure 2.1.1.
The procedure to display a BW image on screen is the following:

- Choose option **Black and White** by mouse.
- Choose option **Display**.

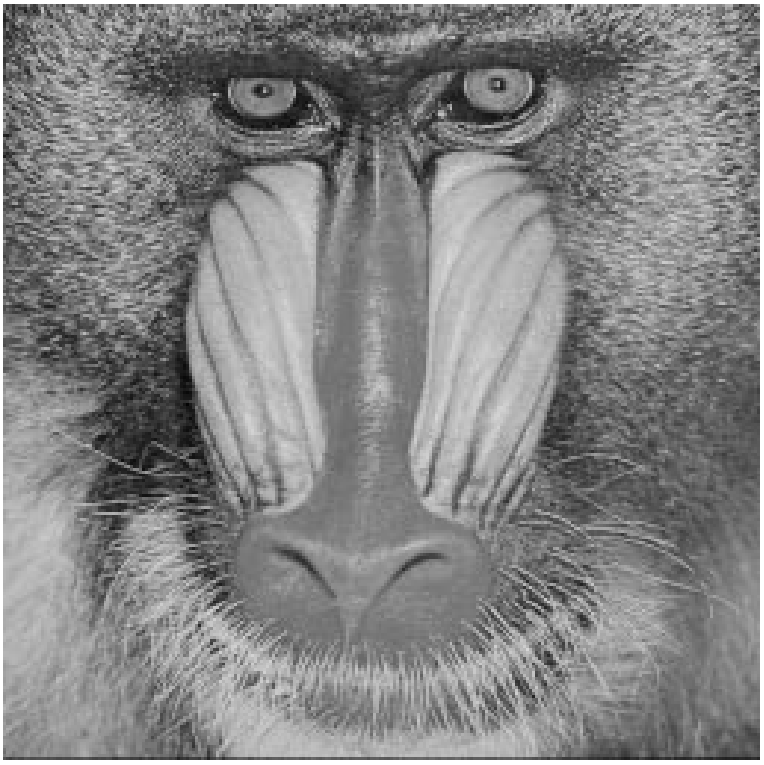

Figure 2.1.1 Grayscale image BABOON.

- A dialog box appears on screen. If the correct image buffer number appears in the box click OK. Otherwise correct Image buffer number and click OK. Let us suppose that buffer 0 is to be displayed. A new window named "BW buffer #0" appears on screen. It can be moved or closed as any other window.

If the image width / height (or both the width and height) exceeds the screen width / height, a scrollbar appears on the image display window. The user can use this scrollbar to view parts of the image not currently on display. Furthermore, the user can scale (zoom in or out) the image using the keyboard buttons "I" and "O" for zoom-in and zoom-out, respectively.

One can easily see the coordinates and gray-level of the various image pixels:

- Choose option **File** by mouse.
- Choose option **Display coord.**.
- A frameless window appears. It contains the x, y coordinates and the corresponding pixel gray level. If you move the mouse within a displayed image window, the coordinate and gray-level values change according to each pixel position. If the cursor is outside any image window, the values are not updated.

This capability is extremely helpful in order to check the image content quickly.

Another important capability is the definition of the region of interest (ROI). It can be done by using the middle mouse button:

- Move the mouse cursor to one corner of the desired ROI.
- Press the middle mouse button and move the cursor to the opposite corner. A rectangle will display the current ROI.
- Release middle mouse button at the desired position.

The ROI can be restored to cover the entire image by pressing the right mouse button inside the image buffer. In this case a verification sound will be heard.

The ROI definition procedure is slightly different when the image is too big to fit into the screen :

- Move the mouse cursor to the upper left corner of the desired ROI.
- Press the middle mouse button.
- Move the mouse cursor to the lower right corner of the desired ROI.
- Press again the middle mouse button.

Again a rectangle appears to display the current ROI.

An image can be saved on a disk file in binary (raw) format as follows:

- Define the part of the image to be saved by selecting a ROI, or select the whole image by clicking the right-hand mouse button inside the image window.
- Choose option **File** by mouse.
- Choose option **Save**.
- A dialog box appears on screen. Fill in the correct image buffer number (e.g. image number 0), choose Raw format from the selection list, and write the OS file name in the corresponding box. The image is now stored on the desired DOS file.
- You can check if the image is properly stored by loading it again, as described previously.

2.2 How to load, display and save a raw color image

In EIKONA for Unix, color image buffers are separate entities from BW image buffers. However, each RGB color buffer consists of three BW buffers. Red component occupies the first buffer, green the second and blue the third one. A special function of the package permits the splitting of a color buffer to its R, G and B components.

Let us suppose that you have a color image consisting of the image files BABOON.r, BABOON.g, BABOON.b of dimensions 256×256 each in binary (raw) format, on your current directory. The corresponding color image

- Choose option **File** by mouse.
- Choose option **Import Raw**.
- A dialog box appears on screen. Click the mouse on any of the three files.
- Give image number 0 in order to load this image file to the image buffer 0. Fill the image width and image height text boxes the value 256. Choose OK. The image is now loaded on image buffer 0.

The color image is automatically displayed on screen.

Color images can be displayed as follows:

- Choose option **Color** by mouse.
- Choose option **Display**.
- A dialog box appears on screen. If the correct Image buffer number appears in the box click OK. Otherwise correct Image buffer number and click OK. Let us suppose that buffer 0 is to be displayed. A new window named "Color buffer #0" appears on screen. It can be moved or closed as any other window.

When our display mode supports only 256 colors, the **Display** routine uses an elaborate way to create a good color palette with 256 colors before displaying the color image. Therefore, it is rather slow. Since in 256 colors display mode EIKONA loads a palette for each image displayed and since only 256 colors can be displayed at once, it is advised that only one color image at a time is displayed. Displaying a color image along with black & white images will also produce strange effects on the displayed images.

We can easily see the coordinates and gray-level values of the various color image pixels:

- Choose option **File** by mouse.
- Choose option **Display coord.**
- A frameless window appears. It displays the x, y coordinates and the pixel R,G,B levels. If you move the mouse within a displayed image window, the coordinates and R,G,B level values change according to each pixel position. Also you can change the displayed color space by right-clicking on the coordinates window. You have the ability to select X,Y,Z or L,a,b or H,L,S color spaces. As you move the mouse within a displayed image window you can see the pixel values in the specified color space. If the cursor is outside any image window, the RGB values are not updated.

The color image buffers can be stored in binary (raw) format on three separate disk files. Usually the R,G,B image buffers that correspond to the color image buffer to be saved are saved on three files having the form *.r, *.g, *.b. For example the BABOON image that is stored on color buffer 0 can be saved in the following way. BW image buffer 0 (R component) can be stored on the file BABNEW.r. Similarly, the BW buffers 1,2 can be stored on the disk files BABNEW.g, BABNEW.b respectively.

2.3 How to load and save images of other formats

In the current version, the TGA image format is supported for 8 and 24 bits per pixel. In order to load a TGA image the option **File, Open, Tga button** must be chosen. Depending on whether the TGA image is a BW or a color one, the chosen buffer number corresponds to a BW or a Color image buffer.

The opposite procedure is followed in order to store an image file on a .TGA disk file. The option **File, Save, Tga button** can be used for this purpose. The user must specify the image buffer to be saved, as well as the image filename. EIKONA saves a grayscale image buffer as a grayscale (8 bit) TGA file, and a color image buffer as a color (24 bit) TGA file. It must be noted that the current version of EIKONA does not support all types of TGA files.

The process to load and store a JPEG image is similar with the above one.

2.4 Examples of grayscale image processing and analysis

Now you are ready to perform some image processing operations. A basic knowledge of digital image processing operations is recommended before proceeding to the rest of the manual. However, this is not a must, especially for simple image processing operations used in producing video effects. For more advanced processing and analysis, it is recommended to refer to the book I. Pitas, *Digital image processing algorithms*, Prentice Hall, 1993. Basic digital image processing literature can be found in the section REFERENCES of this manual. If you have any question on the commands and their operation refer to the second part of this manual by command name. In the rest of this chapter a sequence of menu item selections that have to be performed by user in order to perform a certain task will be given as a sequence of commands in bold, separated by commas. For example the sequence **Black and white, Basic, Copy** means that the user must select **Black and white** from the main menu, then click on **Basic** from the pull-down menu that will show up and finally select **Copy**. Now, you are ready to explore the many exciting facets of image processing.

2.4.1 Basic image processing

Let us suppose that the image file LENNA256.DAT of size 256×256 is stored on BW image buffer 0. We can display it on screen as was already described in the previous section. We assume that all other image buffers are empty (i.e. their pixel values are zero). If we display image buffer 1, a black window will appear on screen.

To brighten LENNA, we can add a suitable constant to the grayscale level of

every point. This can be done by the sequence **Black and white, Basic, Addc**. A dialog box appears. We put 0, 1 as source and destination buffers. The constant is chosen to be 50. The source and destination images are shown in Figure 2.4.1.

Figure 2.4.1 LENNA and its brightened image.

Let us suppose that we want to compute the negative of buffer 0 and store it on buffer 1. This can be done by choosing the functions **Black and white, Processing, Neg**. A dialog box appears on screen. We put 0,1 as source and destination image buffers respectively. At the end of the operation the displayed buffer 1 is repainted showing the negative of image LENNA. The negative image is shown in Figure 2.4.2.

You can copy buffer 1 to buffer 3 by choosing the option **Black and white, Basic, Copy**. A dialog box appears. You can define buffers 1,3 as source and destination buffers respectively.

If we want to threshold an image we can follow the sequence **Black and white, Analysis, Region Segmentation, Threshold**. As an example, if image LENNA is currently loaded in image buffer 0, we can enter 0, 1 as the number of the source and the destination buffers, respectively, and 150 as the threshold level. The original and the thresholded version of LENNA are shown in Figure 2.4.3.

Figure 2.4.2 LENNA and its negative image.

Figure 2.4.3 LENNA and its thresholded image.

Let us suppose that we load images LENNA and BABOON on buffers 0, 1 respectively. If you want to mix the two images of buffers 0, 1 (BABOON and LENNA) and to store the result on buffer 2, you choose the option **Black and white, Basic, Mix**. A dialog box appears. You define buffers 0,1 as source buffers and buffer 2 as destination buffer. You can use 0.5, 0.5 as mixing percentages of the two images. The result can be seen by displaying buffer 2. The mix of the two images is depicted in Figure 2.4.4.

Figure 2.4.4 Result of mixing of LENNA with BABOON.

2.4.2 Image transforms

The Transforms option of the Black and White menu contains various implementations of algorithms concerning transforms of images. The two dimensional FFT is a typical example which we will examine. We load in image buffer 0 the file LENNA256.DAT. We choose **Black and White, Transforms, 2-D FFT** and an input dialog box appears. As the reader can confirm, this dialog box contains float matrix buffers for input. This means that if we had not already allocated at least two float buffers through the **File, Buffers** option, the operation will not function. Assuming that these buffers are allocated, we choose the float buffers where the two parts (real and imaginary) of the Fourier Transform should be stored. After the conclusion of the operation we can select **File, Save, Float Matrix** to retain and further study the results of the transform. We can also test the obtained results through the option **Black and White, Transforms, Inverse 2-D FFT** and see if the resulting image resembles our original one. We could also select **Black and white, Basic, Matrix to Image** to have either the real or the imaginery part of the DFT, converted to image format, in order for us to view it. As an example, let's suppose that the image LENNA is loaded in buffer 0. We can obtain the real and imaginery part of the image, by selecting **Black and white, Transforms, 2-D FFT** and selecting 0 as the source image and 0, 1 as the matrices, where the real and imaginery part of the FFT, will be stored. By selecting **Black and white, Basic, Matrix to Image** and entering 1, 1, we can convert the imaginery part of the FFT of LENNA to image format. If we display buffer 1, the image of Figure 2.4.5 will appear. We must note, though, that by converting a float buffer to an image buffer, effectively the float buffer is clipped between the values of 0 and 255. Thus, the image might not prove to be an accurate representation of the

matrix, in this case.

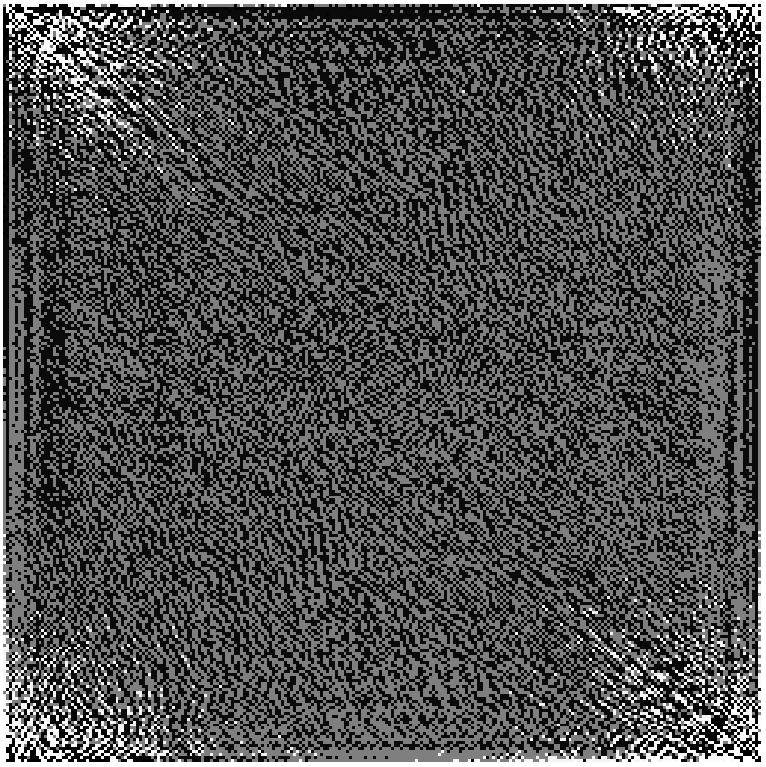

Figure 2.4.5 Imaginery part of the 2-D FFT of LENNA.

The Periodogram, AR PSD and Blackman Tukey PSD are used for Power Spectral Density estimation of images. AR PSD estimation can be easily implemented in EIKONA. Assuming that LENNA is loaded in buffer 0 we can get an AR PSD estimate, by selecting **Black and white, Transforms, AR PSD** and entering 0 as the source image buffer and 0 as the destination float buffer. If we enter 3, 3, and 3 as the AR model left x, right x and y coefficient window sizes, then the result will resemble the one of Figure 2.4.6. We must note that the user should allocate enough float buffers, in order for this operation to complete. Otherwise, EIKONA will complain and will not perform the computation.

If the grayscale image LENNA256.DAT is loaded in image buffer 0, we can get an estimate of the Power Spectral Density by selecting the option **Black and White, Transforms, Periodogram** and entering 0 and 1 as the source and destination image, respectively. The periodogram of LENNA is depicted in Figure 2.4.7. The similarity between the result of this spectral estimation method and of the one of the AR PSD estimate (shown in Figure 2.4.6) is obvious.

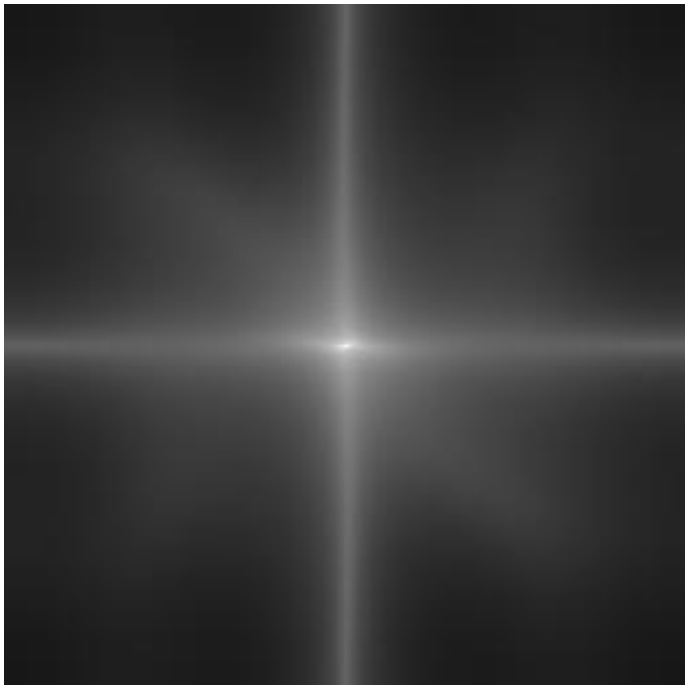

Figure 2.4.6 AR PSD estimate of LENNA.

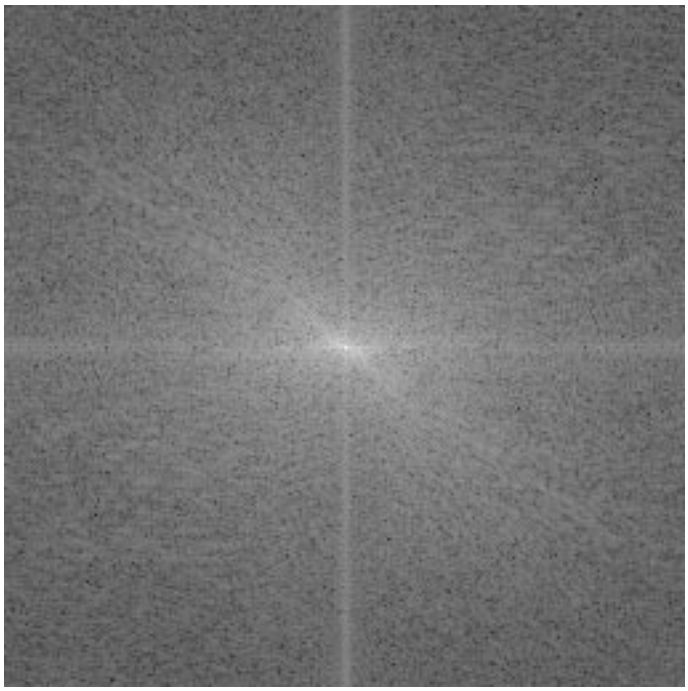

Figure 2.4.7 Periodogram of LENNA.

2.4.3 Digital image filtering and enhancement

EIKONA can be used as a tool to filter out noise from a digital image. Let's take for example the case of an image which is corrupted by impulsive ("salt and pepper") noise [PIT90]. We will assume that the image LENNA is loaded in image buffer 0. You can create a corrupted version of this image by following the sequence **Black and white, Processing, Impulsive**. In the window that appears, you can enter 0, 1 as the number of the source and the destination image, respectively. A commonly used set of values for the noise probability and the minimum and maximum spike values, are 0.1, 0 and 255, respectively. The result can be seen on Figure 2.4.8.

Figure 2.4.8 LENNA and its corrupted version.

Now, you can easily filter out the noise from the corrupted version of LENNA, by utilizing a median filter [PIT90]. This can be done, easily, by selecting **Black and white, Filtering, Median**. If you want to place the resulting filtered image in buffer 2, you should enter 1, 2 as the number of the source and destination images, respectively. The filtering window size is also required. A 3×3 window is usually utilized, so you can fill-in 3, 3 in the remaining window fields. The result is shown in Figure 2.4.9.

As you can see, the filtered image closely resembles the original one. You could experiment with the other noise models and filtering operators that EIKONA supports. For example, using the corrupted image of Figure 2.4.8 as the source image, what would expect for the filtered image to look like, when you utilize a minimum or maximum filter (**Black and white, Filtering** and **Mini** or **Maxi**, respectively)?

Figure 2.4.9 Corrupted and filtered images of LENNA.

Figure 2.4.10 Simulation of photographic film noise.

Simulation of photographic film noise can, also, be performed by using the noise generators provided by EIKONA. The effect shown in Figure 2.4.10 can be created by using the option **Black and white, Processing, Mult_uni** which is a noise generator that creates multiplicative noise of uniform distribution. We use 0, 1 as source and destination buffers and we use noise range 0.3.

In many cases we are interested in improving the visual quality of an image. For example, the contrast of an image can be improved, by performing histogram equalization [PIT90]. In Figure 2.4.11 the histogram for image LENNA, is shown. It can be computed by **Black and white, Analysis, Histogram, Histogram**. If the image of LENNA is in image buffer 0, then enter 0 in the window that appears.

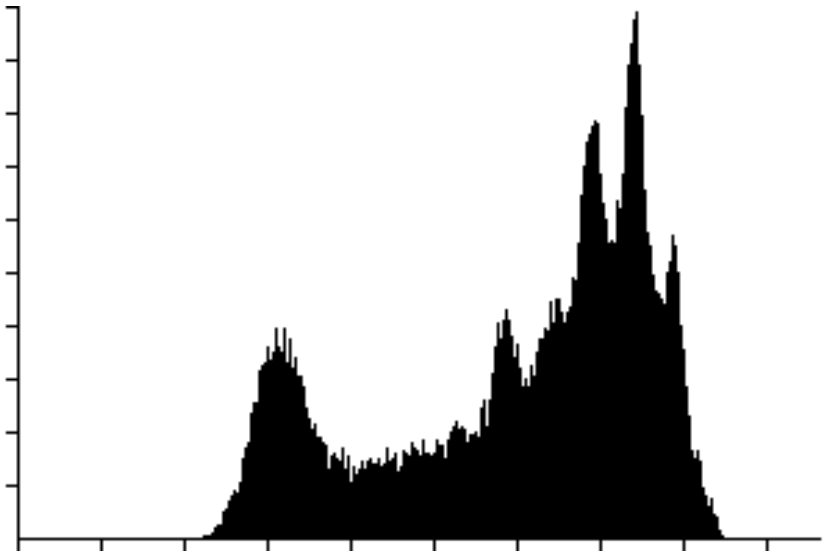

Figure 2.4.11 Histogram of LENNA.

Now, we will perform histogram equalization on LENNA, storing the resulting image on buffer 1. This can be done by the sequence **Black and white, Filtering, Histeq**. Enter 0, 1 as the number of source and destination images, respectively. The resulting histogram-equalized image is shown in Figure 2.4.12, while its histogram is given in Figure 2.4.13.

We can display a Matrix Histogram (pdf or cdf). It can be computed by **Black and white, Analysis, Histogram, Matrix Histogram**, for pdf case and **Black and white, Analysis, Histogram, Matrix Cdf Histogram**, for cdf case.

We can zoom the image buffer 0 and store the results on buffer 1 as follows. We define a ROI on the displayed image buffer 0 that has size less than one quarter of the original LENNA image so that, when zoomed by a factor of 2, the image can fit in the destination buffer. We choose option **Black and white, Processing, Zoom**. A dialog box appears on screen. We put 0, 1 as source and destination image buffers respectively. We use zoom factor 2. We can see the zoomed image

by displaying image buffer 1. The zoomed image is shown in Figure 2.4.14.

Figure 2.4.12 Original and histogram-equalized images of LENNA.

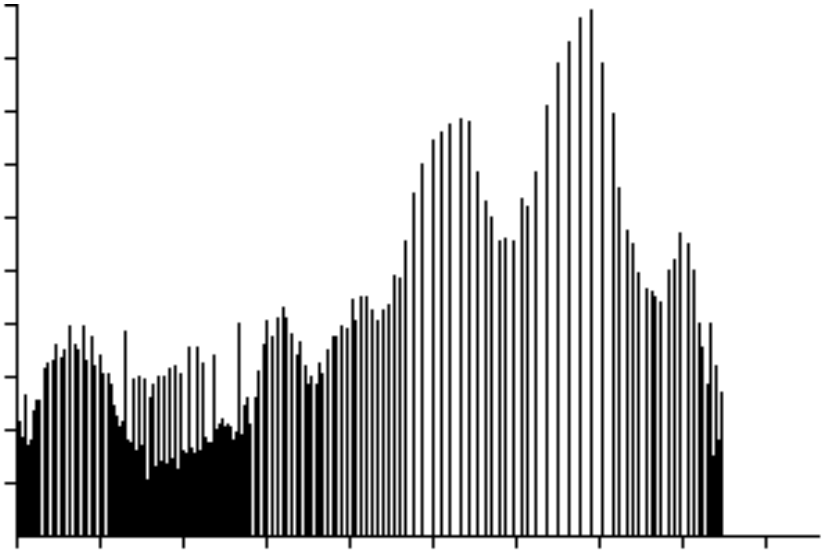

Figure 2.4.13 Histogram of histogram-equalized LENNA.

Figure 2.4.14 LENNA and its zoom version.

EIKONA can be used for image sharpening. You can sharpen part of image of buffer 0, store the results on buffer 1, and display buffer 1 as follows. We will assume that the image BABOON is loaded in image buffer 0. First, copy this image to buffer 1, by selecting **Black and white, Basic, Copy** and entering 0, 1 as source and destination images, respectively. Define by the mouse the ROI to be the left half of image buffer 0. Choose option **Black and white, Processing, Sharp**. Select buffers 0, 1 as source and destination respectively. Display image buffer 1. The result of image sharpening on the half part of the original image is impressive, as it can be seen on screen (Figure 2.4.15).

If you want to sharpen the entire image redefine the ROI to cover the entire image buffer 0. Choose option **Black and white, Processing, Sharp**. Select buffers 0, 1 as source and destination respectively. Display image buffer 1. You can store the sharpened image on a disk file called SHARP256.DAT as follows. Choose option **File, save, Raw button**. Select image buffer 1 and give image file name sharp256.dat.

2.4.4 Nonlinear digital image filtering

EIKONA supports a multitude of nonlinear filtering operations. You can perform many median-based filtering by selecting **Black and white, Non-linear filtering, Median Filters**. If image LENNA is loaded in buffer 0, and you create an image in buffer 1, corrupted by impulsive noise, using the method described in the start of the previous section, you can apply a separable median filter by selecting **Black and white, Non-linear filtering, Median Filters, Separable Median**. Enter 1, 2, 3 as the number of the source, destination and intermediate images, respectively. In the window size fields enter 3, 3. The resulting image, that can be seen by displaying image buffer 2, is depicted in Figure 2.4.16.

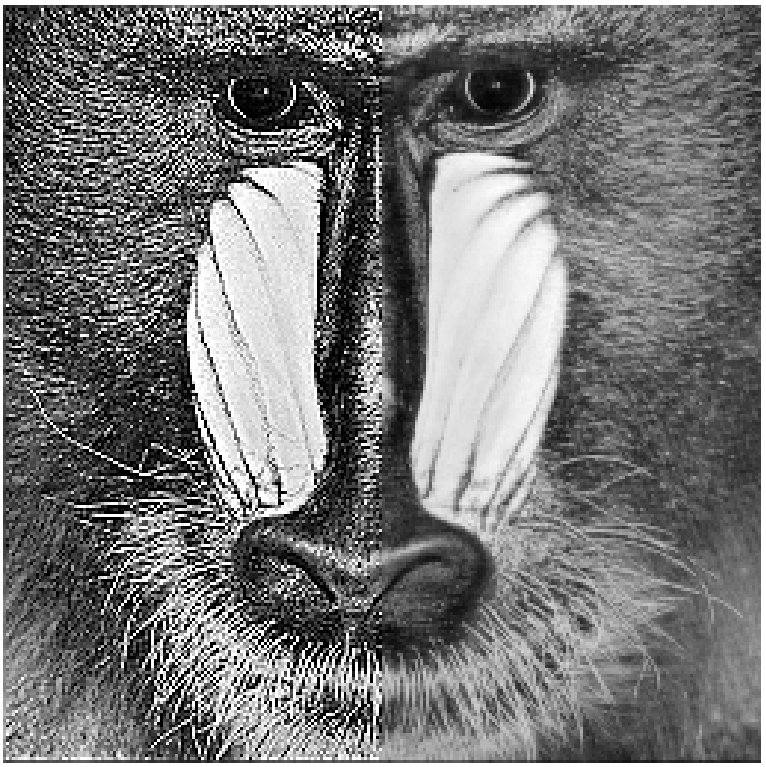

Figure 2.4.15 BABOON with its left side sharpened.

Another non-linear filter is the local adaptive filter, which is based in local statistics. Let us suppose that we have an image of LENNA corrupted by additive uniform noise. This can be done by **Black and white, Processing, Add Uni**. Assuming that LENNA is loaded in image buffer 0, we can enter 0, 1 and 30 in the fields of the window that appears. The corrupted image is shown in Figure 2.4.17.

Now, the local adaptive filter can be utilized to filter out the noise. This can be done by selecting **Black and White, Non-linear filtering, Local adapt. filter**. Enter 1, 2 as the number of the source and the destination images, respectively. Since the noise range, during the creation of the noisy image, was 30, enter 75 as the value of the noise variance. The filtered image can be seen by displaying buffer 2. It is shown in Figure 2.4.18.

Figure 2.4.16 Corrupted and filtered image of LENNA.

Figure 2.4.17 LENNA corrupted by additive uniform noise.

Figure 2.4.18 Corrupted LENNA, filtered by a local adaptive filter.

Many morphological operations are also supported. These include both binary and grayscale morphological operators. Assuming that the thresholded binary image of LENNA, shown in Figure 2.4.3, is loaded in buffer 1, we can perform a binary opening operation, by performing a binary erosion followed by a binary dilation operation. The first step can be performed by selecting **Black and white, Non-linear filtering, Morphology, Binary Erode**. Enter 1, 2 as the number of the source and the destination images, respectively. You can use a cross type structuring element, so enter 2 in the structuring element box. The opened version of the image of Figure 2.4.3 can be obtained by performing a binary dilation operation on the eroded image. Select **Black and white, Non-linear filtering, Morphology, Binary Dilate** and enter 2, 3, 2 as the source, destination images and structuring element type. Both the eroded and the opened images are shown in Figure 2.4.19.

Another well known morphological operation is the skeletonization [PIT93]. It is used to extract the skeleton of an image. EIKONA supports this operation in binary images. Let us suppose that the binary image RECT.BIN is loaded in image buffer 0. Select **Black and white, Non-linear filtering, Morphology, Skeleton**. Enter 0, 1 as the source and destination image and 2 as the parameter. The original image and its skeleton are shown in Figure 2.4.20.

Figure 2.4.19 Eroded and opened images from thresholded LENNA.

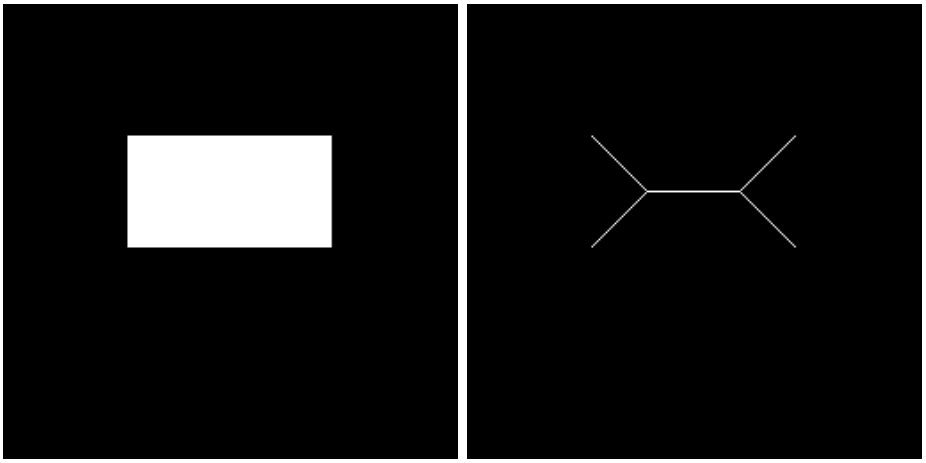

Figure 2.4.20 The binary image RECT and its skeleton.

Small details of a binary image can be extracted by the means of the top hat transformation [PIT93]. Once more, we will use the thresholded image of LENNA of Figure 2.4.3, as a demonstration of the transformation. Assuming that the thresholded LENNA resides in buffer 1, select **Black and white, Non-linear filtering, Morphology, Top Hat** and enter 1, 2, 3, 3 as the values of the source and destination images and window x and y sizes, respectively. The result is depicted in Figure 2.4.21. It is obvious, that this morphological transformation yields many small details of the source binary image.

Figure 2.4.21 Top hat transformation of LENNA.

2.4.5 Edge detection

There are many ways to perform edge detection in an image, using EIKONA. We will show most of these operations, since edge detection is an area of image analysis for which a lot of interest exists.

The compass operation is a directional edge detector. It can detect edges having a specific direction (slope). Although this seems limiting at first, we must take into account that this detector gives us an efficient tool to find edges, we are looking for, with specific slopes. As an example, if LENNA is loaded in buffer 0, we can find lines that are almost vertical, by selecting **Black and white, Analysis, Edge detection, Compass** and entering 0, 1, 90 as the values of the source and destination images and edge direction, respectively. The resulting image is given in Figure 2.4.22.

Figure 2.4.22 Result of the compass edge detector on LENNA.

You can perform edge detection on image buffer 0 and store the results on buffer 2 by choosing the option **Black and white, Analysis, Edge Detection, Sobel** and by selecting 0, 2 as source and destination buffers respectively. Let us suppose that we want to perform edge detection only on the left-hand half of the image. We copy image buffer 0 to image buffer 3 by choosing the option **Black and white, Basic, Copy**. We choose the corresponding ROI by mouse. We can perform edge detection by choosing the option **Black and white, Analysis, Edge Detection, Sobel** and by selecting 0, 3 as source and destination buffers, respectively. The result is shown in Figure 2.4.23.

The edge detector output stored on buffer 2 can be thresholded and its results can be stored on buffer 3 as follows. Restore the ROI to cover the entire image buffer. Choose option **Black and white, Analysis, Region Segmentation, Threshold** and select 2,3 as source and destination buffers. Choose threshold value 70. All pixels having value above 70 take value 1 and the rest take value 0. The resulting image is binary, i.e. it has values 0 and 1. The thresholded edge detector output is shown in Figure 2.4.24.

Figure 2.4.23 Result of the sobel edge detector on LENNA.

Figure 2.4.24 Output of the sobel edge detector on LENNA, after thresholding.

Remaining in the field of Edge Detection, we should give an example of Line Detection. Select option **Black and White, Analysis, Edge Detection, Line Detect**. Assuming that image buffer 0 contains the image LENNA256.DAT, choose image buffer 1 as the output buffer. You should also select a line direction from the allowable range, which is: 0, 45, 90 and 135 degrees. You can check the results of these actions on your monitor. If you enter 0, 1 and 45 as the source and destination images and line direction, then if you display buffer 1, the image of Figure 2.4.25 will appear on screen.

Figure 2.4.25 Result of the line detect operation on LENNA.

Concluding the Edge Detection menu we give a typical example of how to perform Edge Following. Before selecting the option we should first make visible the image buffer we are interested in, using the option **Black and White, Display**. Then we choose the option **Black and White, Analysis, Edge Detection, Edge Following** and after we insert the proper buffer numbers and threshold values (see [PIT93]), a non-modal window appears instructing us to select the start pixel by left clicking the desirable point on the image buffer. We should note that if some windows overlap we can easily move them in order to make properly visible the image buffer of interest.

2.4.6 Region segmentation and texture analysis

We will now explore some applications of EIKONA in the areas of region segmentation and texture analysis.

If the thresholded image of LENNA, of Figure 2.4.3, is loaded in image buffer 0, we can have EIKONA report the number of distinct binary sets that make up the whole image. This can be easily done by selecting **Black and white, Analysis, Region Segmentation, Count items**. Enter 1, 2 as the source and label images. An information window will pop-up, informing us that this binary image of LENNA consists of 82 sets.

EIKONA can also be used to segment a grayscale image to a number of uniform regions. For example, if image LENNA resides in buffer 0, we can segment it to 4 regions and store the resulting grayscale image in image buffer 1, by selecting **Black and white, Analysis, Region Segmentation, Segment**. Enter 0, 1 and 4 in the fields of source/destination images and number of regions, respectively. Figure 2.4.26 depicts the segmented image of LENNA, where each gray scale represents one of each regions.

Figure 2.4.26 LENNA and the segmented resulting image.

We already saw an example of the **Threshold** option, so we move to a description of **Region Grow**. Make visible the input image buffer of interest. Choose **Black and White, Analysis, Region Segmentation, Region Grow**. Select the appropriate input and output image buffers and give the Threshold value which will distinguish one region from another. A non-modal window will appear letting you select the "seeds" (up to 256) before clicking the OK button. As an example, we will perform region grow on LENNA, assuming it is loaded in image buffer 0. Select **Black and White, Analysis, Region Segmentation, Region Grow** and enter 0, 1 as the source and destination images and enter 20 as the threshold value. After selecting some points of the image as seeds, by clicking on them, and pressing "OK", the resulting image might resemble the one of Figure 2.4.27.

The next menu to be described concerns Texture Analysis algorithms. The Gray Level Differences Histogram of an image can be computed in the following

way. Choose option **Black and White, Analysis, Texture, Gray Level Dif. Histogram**. Assuming that image buffer 0 contains the image LENNA, we give 0, 1 as the input and output buffer numbers respectively and enter 1, 1 as the displacement coordinates. Like we saw before, in the Histogram menu, the output is a window containing the visual information of the Gray Level Differences Histogram. It is given in Figure 2.4.28.

Figure 2.4.27 Result of a region grow operation on LENNA.

Some valuable information about this histogram can be calculated through the option **Black and White, Analysis, Texture, Gray Lev. Dif. Hist. Parameters**. Check this information on your monitor in conjunction to the previous example.

One other interesting feature of the Texture Menu is the Angular-Radial option. By choosing **Black and White, Analysis, Texture, Angular-Radial** we can obtain visual information about the angular and radial distribution of the power spectrum of an image in a format similar to the one used to display histograms. For example, the radial and the angular distribution of the power spectrum of LENNA is shown in Figures 2.4.29-2.4.30, respectively.

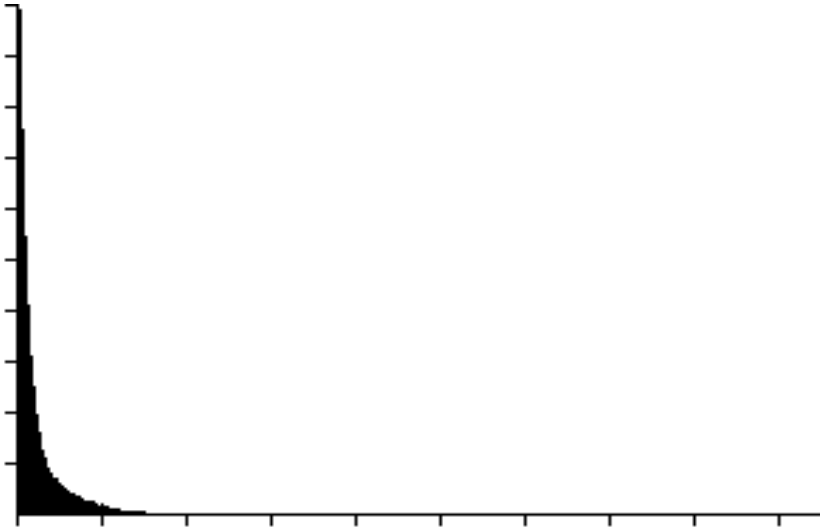

Figure 2.4.28 Gray level dif. histogram of LENNA with displacement values 1,1.

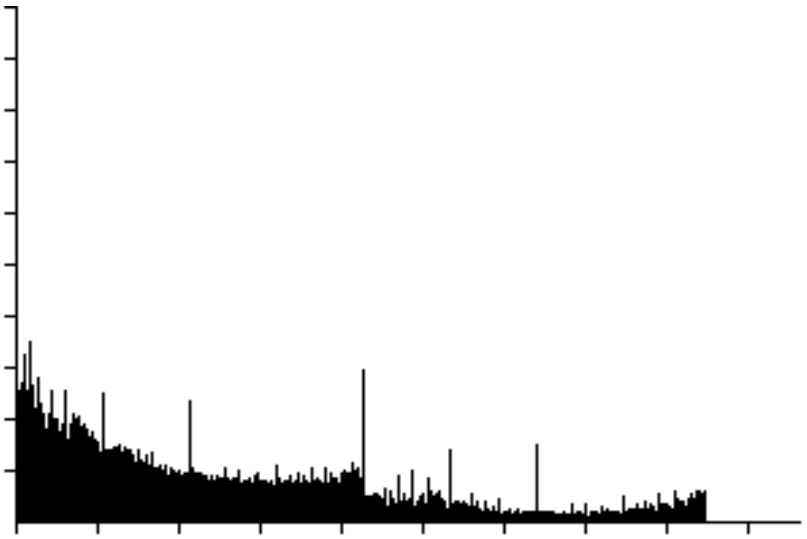

Figure 2.4.29 Radial distribution of PSD of LENNA.

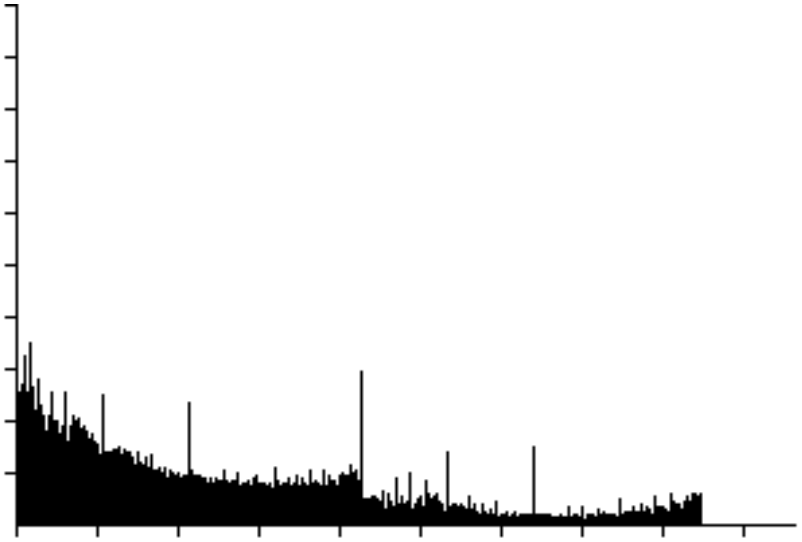

Figure 2.4.30 Angular distribution of PSD of LENNA.

2.4.7 Shape description

The Shape Analysis menu gives valuable information about objects of grayscale and binary images. We should remind you that any grayscale image can be transformed to a binary one by using the **Black and White, Analysis, Region Segmentation, Threshold** option and giving the desired number to act as a threshold between 0 and 1.

We can obtain the characteristics of a binary image by selecting **Black and White, Analysis, Shape Analysis, Find Char**. Let us suppose that image RECT.BIN is loaded in image buffer 0. By selecting the operation mentioned above, and entering 0, 0 as the source image and the threshold value, EIKONA will respond with a pop-up window, which gives information regarding the the characteristics of the first object to meet in a row wise manner from the upper left corner (i.e. the rectangle). The reader must have understood by now that in order to choose another object, he should adjust the ROI appropriately. For our second example the above mentioned ROI adjustments around an object of interest also hold. Choose the ROI to be the area surrounding the rectangle. If you choose the option **Black and White, Analysis, Shape Analysis, Chain Coding** and select image buffers 0,1 as input and output and a moderate number for the maximum number of chain nodes (in this case 2000), you can check the results on your monitor. It is obvious that Chain Coding can be used as another Edge Detection algorithm for binary images. Figure 2.4.31 depicts the result of chain coding.

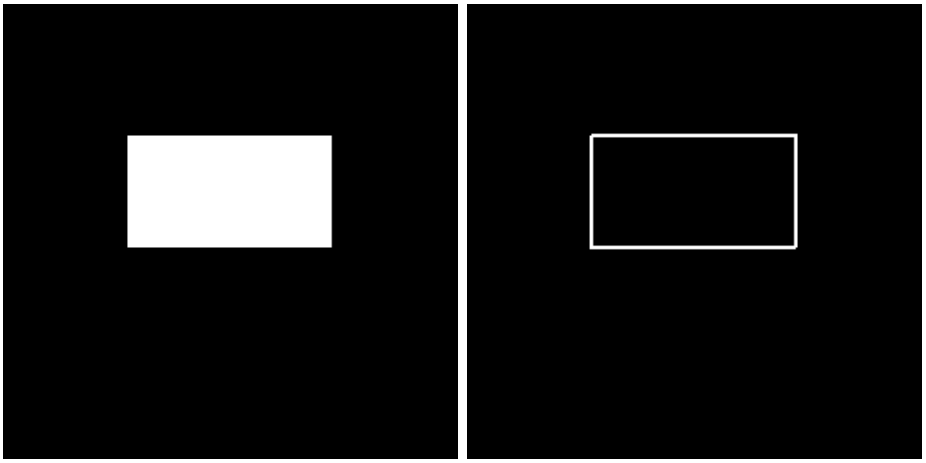

Figure 2.4.31 Original and chain coded image of RECT.

Figure 2.4.32 Thresholded LENNA and the result of one-pass thinning.

Concluding this section we will describe the Pyramid and Thinning options. If we choose **Black and White, Analysis, Thinning** we come up with a submenu containing two options: **One Pass Thinning** and **Two Pass Thinning**. We should note that these are the only options having as output image buffer the input image buffer. As a result, we must be very careful when using these two options, because our original image can be destroyed. If we perform one-pass thining on the thresholded image of LENNA of Figure 2.4.3, then the binary image of Figure 2.4.32 will appear on screen after displaying image buffer 0.

By choosing **Black and White, Analysis, Pyramid** and giving the appropriate input and output image buffers, we come up with a very impressive result, as you can verify using the image LENNA256.DAT. It is obvious that adjusting correctly the ROI around images and copying or saving, can give us results similar to those obtained by using the **Black and White, Processing, Decim** option. Assuming that the thresholded image of LENNA (see Figure 2.4.3) is loaded on buffer 0, by selecting **Black and White, Analysis, Pyramid** and entering 0, 1 as the source and destination images, the image stored in image buffer 1 will resemble the one of Figure 2.4.33.

Figure 2.4.33 Pyramid of thresholded LENNA.

Any binary image can be printed on a Hewlett-Packard compatible laser printer as follows. We choose option **File, Print to File, HP Laserjet**. We give buffer number 3, DPI 300 and output file PARTHP. This command produces the disk file PARTHP as output. It can be printed by any Unix print command.

Binary images can also be printed from within the EIKONA environment by issuing the commands **File, Print** In the case of **HP Laserjet** we give the number of the source image buffer (e.g. 3) and the dpi (e.g. 300).

In this section some examples of grayscale and binary image analysis routines

have been given. EIKONA contains several powerful routines for image analysis (e.g. mathematical morphology) that are a very useful tool for the interested and/or advanced user. Information on how to use these routines and commands are found in the library reference manual.

2.5 Examples of color image processing and analysis

In this section we will examine some of the operations on color images, that EIKONA supports. With some notable exceptions, these operations replicate the ones used in the section of grayscale image processing and analysis. For this reason, only a small subset of the operations will be covered here.

2.5.1 Basic image processing

We can easily copy a color image to another by selecting **Color, Basic, Copy**. We must enter the source and the destination image buffer number in the box that appears.

We can also add a constant value to every one of the red, green and blue color components that compose the color image. This can be done by selecting **Color, Basic, Addc**. In the box that appears enter the source and destination color image buffer, as well as the constant to be added.

2.5.2 Color transforms

EIKONA implements a large number of color transformations. The CMY transformation essentially produces the "negative" of a color image. It can be performed as follows. We restore ROI to cover the entire image. We choose the option **Color, Color repr., From RGB to, CMY**. We give the source and destination color buffer numbers 0,1 respectively. The negative can be seen by displaying buffer 0. The resulting image is depicted in Figure 2.5.1. The negative image can be transformed to positive again by choosing the option **Color, Color repr., To RGB from, CMY** and given 1,1 as source and destination buffers. Color image buffer 0 contains now the "positive" image.

2.5.3 Digital image filtering and enhancement

Let us suppose that the image file BABOON.TGA is stored on color image buffer 0. The image size is 256×256 pixels. We can display it on screen as was already

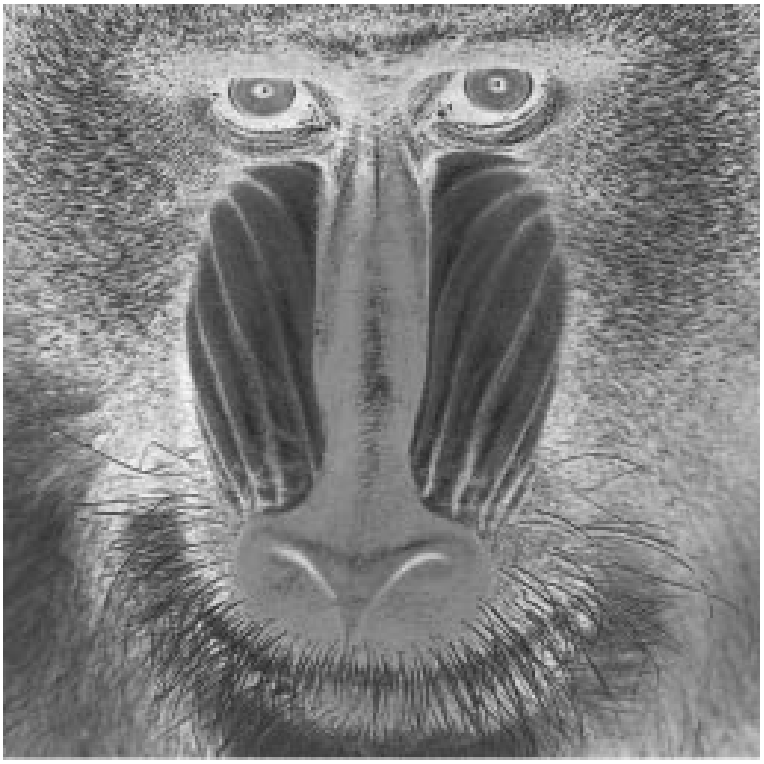

Figure 2.5.1 CMY color model image of BABOON.

described in a previous section. We assume that all other image buffers are empty (i.e. their pixel values are zero).

We can zoom the image buffer 0 and store the results on buffer 1 as follows. We define a ROI on the displayed image buffer 0 that has size less than one quarter of the original LENNA image. We choose option **Color, Basic, Zoom**. A dialog box appears on screen. We put 0,1 as source and destination image buffers respectively. We use zoom factor 2. We can see the zoomed image by displaying image buffer 1 and is depicted in Figure 2.5.2. We clear the color image buffer 1.

We can shorten (decimate) the color image buffer 0 and store the results on buffer 1 as follows. We redefine the ROI on the displayed image buffer 0 so that it covers the entire buffer by clicking the right-hand mouse button outside an displayed image window. We choose option **Color, Basic, Decim**. A dialog box appears on screen. We put 0,1 as source and destination image buffers respectively. We use decimation factor 2. We can see the decimated image by displaying image buffer 1.

Simulation of photographic film noise can be performed by using the noise generators provided by EIKONA. The effect shown in Figure 2.4.11 can be created by using the option **Color, Processing, Mult_uni** which is a noise generator that creates multiplicative noise of uniform distribution. We use 0,1 as source and destination buffers and we use noise range 1.0. This effect is particularly evident

Figure 2.5.2 Zoomed image of BABOON.

in the bright parts of the image. The resulting image is given in Figure 2.5.3.

You can store the corrupted image on a disk file called SBAB.TGA as follows. Choose option **File, Save, Tga**. Select color image buffer 1 and give image file name sbab.tga. If you want to delete this file, you can switch to a Unix shell and you can delete the file by a Unix command.

Histogram equalization can be performed as follows. Let us suppose that the image BABOON.TGA is stored on buffer 0. The histogram of its R,G,B components can be calculated by choosing option **Black and white, Analysis, Histogram** and by giving the BW image number 0,1,2 respectively. The image histogram is used implicitly to perform image enhancement by histogram equalization. Select **Color, Processing, Histeq**. Enter 0, 1 as the source and destination color buffers, respectively. The resulting histogram-equalized image is given in Figure 2.5.4.

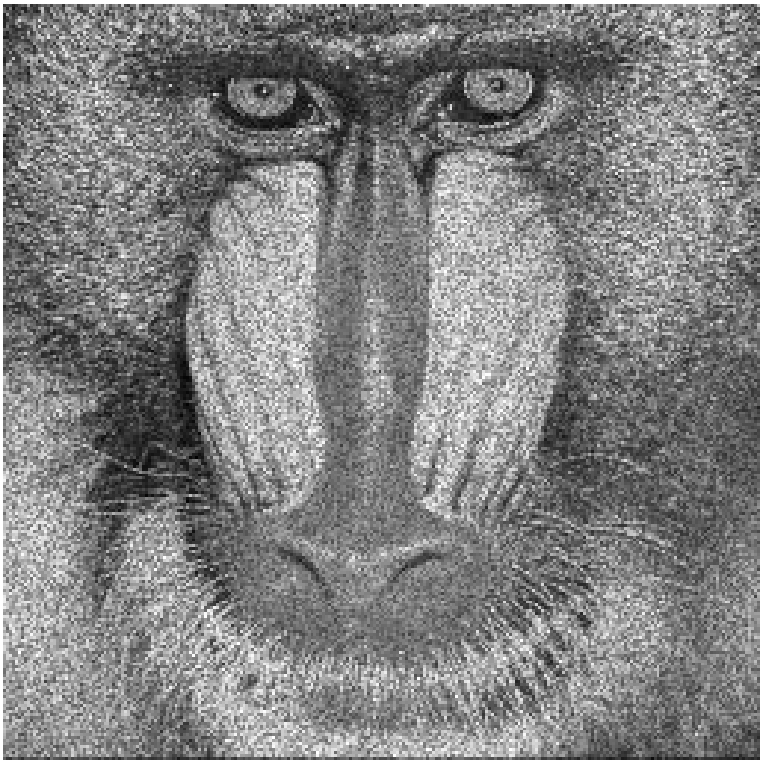

Figure 2.5.3 Simulation of photographic film noise on color image BABOON.

2.5.4 Edge detection

You can perform edge detection on color image buffer 0 and store the results on buffer 1 by choosing the option **Color, Analysis, Sobel** and by selecting 0,1 as source and destination buffers respectively. We copy image buffer 0 to image buffer 1 by choosing the option **Color, Basic, Copy**.

Let us suppose that we want to perform edge detection only on the left-hand half of the image. We choose the corresponding ROI by mouse. We can perform edge detection by choosing the option **Color, Analysis, Sobel** and by selecting 0,1 as source and destination buffers respectively. The result is impressive, as it can be seen in Figure 2.5.5.

2.5.5 Region segmentation and texture analysis

Region segmentation can be performed by using the option **Color, Analysis, Segment** and by choosing 0,1 as source and destination buffers. We choose number of regions equal to 4. The result can be seen in Figure 2.5.6.

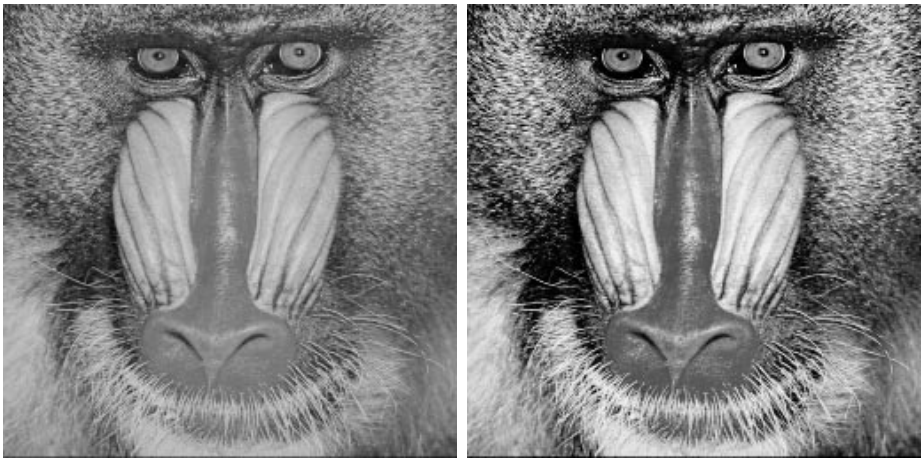

Figure 2.5.4 Original and histogram-equalized images of BABOON.

Figure 2.5.5 Sobel edge detection on image BABOON.

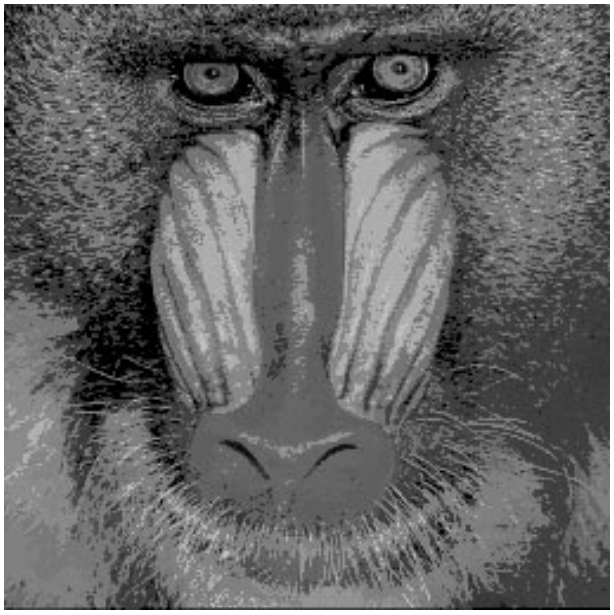

Figure 2.5.6 Segmentation of image BABOON.

References

[PIT93] I.Pitas *Digital image processing algorithms*, Prentice Hall, 1993.

[PIT93b] I.Pitas, editor, *Parallel algorithms for digital image processing, computer vision and neural networks*, J. Wiley, 1993.

[PIT90] I.Pitas, A.N.Venetsanopoulos, *Nonlinear digital filters: Principles and applications*, Kluwer Academic, 1990.

[AND77] H.C.Andrews, B.R.Hunt, *Digital image restoration*, Prentice Hall, 1977.

[ANG90] E.Angel, *Computer graphics*, Addison-Wesley, 1990.

[BAL82] D.H.Ballard, C.M.Brown, *Computer vision*, Prentice Hall, 1982.

[FOL90] J.D.Foley, A.van Dam, S.K.Feiner, J.F.Hughes, *Computers graphics: Principles and practice*, Addison-Wesley, 1990.

[GON87] R.C.Gonzalez, P.Wintz, *Digital image processing*, Addison-Wesley, 1987.

[HAR87] S.Harrington, *Computer graphics: A programming approach*, McGraw-Hill, 1987.

[JAI89] A.K.Jain, *Fundamentals of digital image processing*, Prentice Hall, 1989.

[LEV85] M.D.Levine, *Vision in man and machine*, McGraw-Hill, 1985.

[LIN91] C.A.Lindley, *Practical image processing in C*, Wiley, 1991.

[MIC87] *Microsoft C: Runtime library reference*, Microsoft Press, 1987.

[NIB86] W.Niblack, *Digital image processing*, Prentice Hall, 1986.

[PRA91] W.K.Pratt, *Digital image processing*, Wiley, 1991.

[PRE88] W.H.Press, B.P.Flannery, S.A.Teukolsky, W.T.Vetterling, *Numerical recipes in C*, Cambridge University Press, 1988.

[RIM90] S.Rimmer, *Bit-mapped graphics*, Windcrest, 1990.

[ROS82] A.Rosenfeld, A.C.Kak, *Digital picture processing*, Academic Press, 1982.

[SCH89] R.J.Schalkof, *Digital image processing and computer vision*, Wiley, 1989.

[SER82] J.Serra, *Image analysis and mathematical morphology*, Academic Press, 1982.

[WIL87] R.Wilto, *Programmer's guide to PC and PS/2 video systems*, Microsoft Press, 1987.

[WYZ67] G.W.Wyzecki, W.S.Stiles, *Color science*, Wiley, 1967.

Index