YILDIZ TECHNICAL UNIVERSITY
ELECTRICAL-ELECTRONIC FACULTY
COMPUTER ENGINEERING DEPARTMENT

SENIOR PROJECT

Image Processing Tool

Project Manager: Asist. Prof. Sırma YAVUZ

Project Group

03011018 Akın MOĞULKOÇ

03011030 Orçun ATAY

İstanbul, 2007

CONTENTS

ABBREVIATION LIST:

CPU	Central Processing Unit
BMP	Bit Map Format
JPEG	Joint Photographic Experts Group
PNG	Portable Network Graphics
WBMP	Wireless Bitmap Format
GIF	Graphics Interchange Format
TIFF	Tagged Image File Format
API	Application programming interface
RGB	Red Green Blue Space
HSL	Hue Saturation Luminance Space

FIGURE LIST

TABLE LIST

PREFACE:

Nowadays, increasing image processing applications, and supporting applications for the development of these applications are gaining importance. Likewise, the image processing tool we developed gathers the commonly used image processing techniques. In this way, this tool helps not only the image processing projects but also useful for obtaining sample images. Thus, the tool aims to decrease the project processing durations.

In any project, even while some simple processes are applied on any image too much time may be lost, which is not the main aim of the project. That is for what the application we developed is useful.

Special thanks to our consultant teacher Assist. Prof. Sırma YAVUZ for guiding and standing us.

ÖZET:

JAVA dili ile platform bağımsız olarak tasarlanan ve gerçeklenen uygulamamız 40'ı aşkın işlem seçeneği ve kullanışlı ara yüzü ile görüntü işleme konusundaki çalışmalara etkin bir yardımcı özelliği taşımaktadır. Bahsettiğimiz 40'ı aşkın işlev için parametreleri kullanıcıdan alma yoluyla kullanım alanı genişletilmiş ancak bu projenin hedef kullanıcı kitlesi için belirli bir bilgi birikimi gerekliliğini ortaya koymuştur. Bu sebeple uygulamamız sıradan bir resim düzenleme aracından farklılaşmıştır.

Geliştirdiğimiz uygulama RGB ve HSL renk uzayı için renk filtreleri, matematik ve morfolojik işlemler, konvolüsyon ve smooting işlemleri, fourier, vb gibi işlemleri kolaylıkla gerçekleştirebilecek yapıya ve kullanıcı dostu ara yüze sahiptir.

ABSTRACT:

The platform independent application we designed and realized by JAVA language has a range of processing options of over 40 and a user-friendly interface, and is a very useful tool for image processing applications. For these 40 processing options, the parameters are entered by the users, and in this way the use area has spread, and this situation resulted in the necessity that the target users should have certain knowledge about the application we developed.

The application we developed has the capability to handle the processes such that the color filters for RGB and HSL color space, mathematical and morphological processes, convolution and smoothing processes, Fourier etc. This application also has a user-friendly interface to realize the processes mentioned above easily and comfortably.

1. INTRODUCTION

Many of the techniques of digital image processing, or digital picture processing as it was often called were developed in 1960s with the applications such that satellite imagery, medical imaging, videophone, character recognition, and photo enhancement. However, the cost of processing was fairly high with the computing equipments of that era. In this era this processes could only be carried out by the scientists with powerful computers, but now image processing can be done by anyone with a personal computer or workstation. This is because of the constant price reduction in computers, the increase in CPU speed and graphics power. This increase in accessibility makes image processing an indispensable part of the computer science and the commercial software's.

2. SYSTEM ANALYSIS

2.1. The Problem

With the increasing number of image processing projects; pre-processing has become important. During developing image processing applications; implementing pre-processing techniques, although some of them like basic morphological operations and convolutions filters seems easy to achieve, within the application itself wastes developers' time and increases project costs. For example in a character recognition application, to get better results, images should be pre-processed with operations like cleaning, smoothing etc which dramatically slows down the application.

2.2. Solution

Because of the high human resources costs, third party image pre-processing tools or libraries are preferred for reducing the development period. Image processing tool to be developed in this project will provide basic pre-processing operations independently to the developer. With the use of this tool, developers can easily obtain suitable sample images for testing their applications. Some of the functions the tool includes:

- Color filters
- HSL filters
- Binarization filters
- Automatic binarization
- Mathematical morphology filters
- Convolution filters
- 2 Source filters
- Edge detectors
- Blob counter
- Connected components labeling
- Pixellate, Simple skeletonization, Jitter, Shrink, Oil painting
- Levels linear filter, gamma correction
- Median filter, Adaptive smoothing, Conservative smoothing
- Resize and Rotate

- Fourier transformation (low pass and high pass filters)

The tool will be able to read and write several image file formats. These are:

- BMP: Bit Map Format
- JPG, JPEG: Joint Photographic Experts Group
- PNG: Portable Network Graphics
- WBMP: Wireless Bitmap Format
- GIF: Graphics Interchange Format
- TIFF : Tagged Image File Format

2.3 Feasibility Analysis

2.3.1 Technical Feasibility

2.3.1.1. Hardware

Our image processing tool will be developed to run on average PC configuration, however, when we carry out matrix operations for manipulating pixel data, it is obvious that we need CPU power and more memory. Considering these requirements, recommended configuration is given on Table 2.1.

CPU	P4 or Higher
Main Memory	512 MB
Graphic Memory	64 MB
HDD	40 MB

Table 2.1-Recommended Configuration

2.3.1.2. Software

When we confuse the implementation level of the project we need a complex, fast and useful programming language and we have alternatives like JAVA, .NET, C++ etc. JAVA and .NET are more useful for image processing but C++ is a better solution in point of speed.

Entire system will be developed using JAVA 6 platform. Platform independent, secure, object oriented structure properties of Java and facilities are the most important reasons for this choice. On the other hand claim of JAVA 6 about the speed makes it a better choice then other alternatives.

2.3.2. Economic Feasibility

2.3.2.1 Hardware Costs

According to exigencies which are exhibited in technical feasibility we need two PCs for development duration. The detailed configuration and the costs can be seen on Table 2.2.

Hardware	Product	Price	Quantity
CPU	Intel P4 or AMD	80$	2
Main Board	Asus P4S800D	50$	2
HDD	40GB	40$	2
Main Memory	512 MB	50$	2
Graphic Card	64 MB	35$	2
Monitor	17" Standard	110$	2
Case	Midi Towel	25$	2
Optical Driver	52X	20$	2
Keyboard	Standard	5$	2
Mouse	Standard	3$	2
TOTAL		**836$**	

Table 2.2 Configuration Details

2.3.2.2 Software Costs

All of the programs, which we will use during the development, are open source products and they can be downloaded from their respective official web sites for free.

- Java 1.6 Standard Edition (www.sun.com)
- NetBeans 5.5 IDE (www.netbeans.org)

For this project no extra API's or libraries will be used and the design will be implemented using the standard JAVA packages.

2.3.2.3 Total Costs

Hardware	**836$**
Software	**0$**
Human Resource	**2000$**
TOTAL	**2836$**

Table 2.3 Total project costs

According to the Table 2.3 total costs of the project is 2836$. However, it is a considerably little amount when we consider the gain of development time for this kind of a specific image processing project.

3. FUNCTIONS OF PROJECT:

3.1. Grayscale

This mode uses up to 256 shades of gray. Every pixel of a grayscale image has a brightness value ranging from 0 (black) to 255 (white). Grayscale values can also be measured as percentages of black ink coverage (0% is equal to white, 100% to black). Images produced using black-and-white or grayscale scanners typically are displayed in Grayscale mode.

Although Grayscale is a standard color model, the exact range of grays represented can vary, depending on the printing conditions.
y = 0.299 * R + 0.587 * G + 0.114 * B

3.2. Sepia

The Toning filter has many uses other than toning black and white photographs. It will act on any RGB image. Sepia filter allows you to give that yellow tint to your pictures, as if they were shot near the end of the day. This filter will yield a pleasant Antique effect. The algorithm which is used in sepia filter is:

```
gry = (r + g + b) / 3;
r = g = b = gry;
r = r + (depth * 2);
g = g + depth;
```

If red or green value is over than 255 you accept those 255.

3.3. Invert

Inverting the colors of the image creates a photo-negative effect. It is very simple to invert an image:
R = 255 – R
G = 255 – G
B = 255 – B

3.4. Rotate

Rotating is turning the image with the given angle. When image is turned it does not fit in the old rectangular. If image is located in the old rectangular some data will be lost. Because of this the size of the rectangular that contains the image must be increased.

3.5. Channel Extraction

Once an image with solely the blue and red cells is obtained, the blue and red cells need to be separated from one another. This is accomplished with a simple loop where each pixel in the image that has a red channel value greater than the blue channel value is considered the color red and vice versa for a blue colored pixel. The binary templates from the green channel extraction, background extraction, and red/blue extraction are intersected together so that only the blue or red pixels of the image are passed.

This filter gives all pixels red, green, and blue values the same value. So the image becomes a grayscale image. If red channel is being extracted for all pixels the red value is given to green and blue values. Then the image is grayscale and the gray value is equal to old red value. Green extraction and blue extraction are same; the green or blue value becomes the gray value.

3.6. Channel Filtering

This filter gives each pixel in your image a new color: the color channel that contributes the most to the color of a pixel is removed (except for gray pixels, which are

kept gray). For red, green, and blue values maximum and minimum values are taken from the user. Then, for all pixels if red is not between red-max and red-min values the red value is set 0. It is same for green and blue values.

3.7. Color Filtering

Color filtering is like channel filtering but there is a small different detail. In this filter if red is not between red-max and red-min then red, green, and blue values are set 0. The same operation is done for green and blue values too.

3.8. HSL Filtering

Based on the human perception of color, the HSL model describes three fundamental characteristics of color:

• Hue is the color reflected from or transmitted through an object. It is measured as a location on the standard color wheel, expressed as a degree between 0° and 360°. In common use, hue is identified by the name of the color such as red, orange, or green.

• Saturation, sometimes called chroma, is the strength or purity of the color. Saturation represents the amount of gray in proportion to the hue, measured as a percentage from 0% (gray) to 100% (fully saturated). On the standard color wheel, saturation increases from the center to the edge.

• Lightness is the relative brightness or darkness of the color, usually measured as a percentage from 0% (black) to 100% (white).

HSL filtering is so similar to color filtering. Just the values are not red, green, and blue; they are Hue, Saturation, and Lightness. If one of hue, saturation or lightness value is not between the given maximum and minimum values then for that pixel all of them are set 0.

3.9. Binarization Filters:

When you need to binarize a gray-levels image, you need to select a threshold value, because gray values ranges from 0 to 255: pixels lower than these values are considered background (paper) and pixels higher than this value are considered foreground (text and graphic elements).

3.9.1. Threshold: In this method the pixel values under the given threshold value are accepted white and the others are accepted black.

3.9.2. Threshold with Cary: In this method the threshold value is not given by the user; it is calculated from the image pixel color values by iteration. The method has 5 steps:

1. A threshold value (T) is chosen or calculated from the histogram of the image.
2. According to T the pixel values are separated into two groups.
3. For each group average values M1 and M2 are calculated.
4. New threshold value: T = (M1 + M2)/2
5. These 4 steps are repeated until the difference between the two T values is small enough.

3.9.3. Ordered Dithering: We can express the patterns in compact form as the order of dots added:

8 3 4	and	1 7 4
6 1 2		5 8 3

7 5 9 6 2 9

Then we can simply use the value in the array as a threshold. If the value of the pixel (scaled into the 0-9 range) is less than the number in the corresponding cell of the matrix, plot that pixel black, otherwise, plot it white. This process is called ordered dither. As before, clustered patterns should be used for devices which blur dots. In fact, the clustered pattern ordered dither is the process used by most newspapers, and the term half toning refers to this method if not otherwise qualified.

3.9.4. Bayer Dithering: Bayer has shown that for matrices of orders which are powers of two there is an optimal pattern of dispersed dots which results in the pattern noise being as high-frequency as possible. The pattern for a 2x2 and 4x4 matrices are as follows:

```
1 3      1  9  3 11      These patterns (and their rotations
4 2     13  5 15  7       and reflections) are optimal for a
        4 12  2 10       dispersed-pattern ordered dither.
       16  8 14  6
```

Bayer's method is in very common use and is easily identified by the cross-hatch pattern artifacts it produces in the resulting display. This artifact is the major drawback of the technique which is otherwise very fast and powerful. Ordered dithering also performs very badly on images which have already been dithered to some extent. As stated earlier, dithering should be the last stage in producing a physical display from a digitally stored image. The dithered image should never be stored itself.

3.9.5. Floyd-Steinberg Dithering:

This technique is a modified threshold operation. The algorithm proceeds by moving left-to-right and top-to-bottom over each pixel. Each pixel is threshold on the value 128 for 8-bit images. An error-diffusion step is then performed where the "error" between the input and output pixel is propagated to neighboring pixels according to a fixed system. The error E at pixel P is the quantity (P-threshold (P)). The error E is distributed such that the pixel to the immediate east of P is incremented by 7E/16, the pixel to the south-west of P is incremented by 3E/16, the pixel to the south of P is incremented by 5E/16 and the pixel to the south-east of P is incremented by E/16. Once neighboring pixels have been adjusted, the threshold operation continues in left-right-top-bottom fashion. Figure 4 gives an example of using Floyd-Steinberg diffusion to dither a gray-scale image.

3.9.6. The Burkes Dithering:

Daniel Burkes of TerraVision undertook to improve upon the Stucki filter in 1988:

```
      *  8  4          The Burkes filter
2  4  8  4  2  (1/32)
```

Notice that this is just a simplification of the Stucki filter with the bottom row removed. The main improvement is that the divisor is now 32, which allows the error values to be calculated using shifts once more, and the number of neighbors communicated with has been reduced to seven. Furthermore, the removal of one row

reduces the memory requirements of the filter by eliminating the second forward array which would otherwise be needed.

3.9.7. The Jarvis, Judice, and Ninke filter

If the false Floyd-Steinberg filter fails because the error is not distributed well enough, then it follows that a filter with a wider distribution would be better. This is exactly what Jarvis, Judice, and Ninke did in 1976 with their filter:

```
        *  7  5
3  5  7  5  3
1  3  5  3  1  (1/48)
```

While producing nicer output than Floyd-Steinberg, this filter is much slower to implement. With the divisor of 48, we can no longer use bit-shifting to calculate the weights but must invoke actual DIV (divide) processor instructions. This is further exacerbated by the fact that the filter must communicate with 12 neighbors; three times as many in the Floyd-Steinberg filter. Furthermore, with the errors distributed over three lines, this means that the program must keep two forward error arrays, which requires extra memory and time for processing.

3.9.8. The Sierra filters:

In 1989, Frankie Sierra came out with his three-line filter:

```
*  5  3              The Sierra3 filter
2  4  5  4  2
   2  3  2     (1/32)
```

A year later, Sierra followed up with a two-line modification:

```
       *  4  3              The Sierra2 filter
1  2  3  2  1  (1/16)
```

and a very simple "Filter Lite," as he calls it:

```
     *  2                   The Sierra-2-4A filter
1  1          (1/4)
```

Even this very simple filter, according to Sierra, produces better results than the original Floyd-Steinberg filter.

3.9.9. The Stucki filter:

P. Stucki offered a rework of the Jarvis, Judice, and Ninke filter in 1981:

```
       *  8  4
2  4  8  4  2
1  2  4  2  1  (1/42)
```

Once again, division by 42 is quite slow to calculate (requiring DIVs). However, after the initial 8/42 is calculated, some time can be saved by producing the remaining fractions by shifts. The Stucki filter has been observed to give very clean, sharp output, which helps to offset the slow processing time.

3.10. Mathematical and Morphologic Filters:

In the processing and analysis of images it is important to be able to extract features, describe shapes and recognize patterns. Such tasks refer to geometrical

concepts such as size, shape, and orientation. Mathematical morphology uses concepts from set theory, geometry and topology to analyze geometrical structures in an image.

3.10.1. Erosion:
Classical erosion that removes pixels based on nearest neighbor comparisons always produces distortion in shapes when applied more than a few times, because one pixel removed in the diagonal direction corresponds to a distance of 1.414 (square root of 2, the diagonal of a square) pixels compared to removing one in a vertical or horizontal direction.

3.10.2. Dilatation:
It is vice versa of the erosion. This method enlarges the image and makes it thicker.

3.10.3. Opening:
The Opening block performs an erosion operation followed by a dilation operation using a predefined neighborhood or structuring element.

3.10.4. Closing:
The Closing block performs a dilation operation followed by an erosion operation using a predefined neighborhood or structuring element.

3.10.5. Hit and Miss Thinning:
The binary hit-or-miss transform is applied to filter digital gray-scale signals. This is accomplished by applying a union of hit-or-miss transforms to an observed signal's umbra and then taking the surface of the filtered umbra as the estimate of the ideal signal. The hit-or-miss union is constructed to provide the optimal mean-absolute-error filter for both the ideal signal and its umbra. The method is developed in detail for thinning hit-or-miss filters and applies at once to the dual thickening filters. It requires the output of the umbra filter to be an umbra, which in general is not true. A key aspect of the paper is the complete characterization of umbra-preserving union-of-hit-or-miss thinning and thickening filters. Taken together, the mean-absolute-error theory and the umbra-preservation characterization provide a full characterization of binary hit-or-miss filtering as applied to digital gray-scale signals. The theory is at once applicable to hit-or-miss filtering of digital gray-scale signals via the three- dimensional binary hit-or-miss transform.

3.11. Convolution Filters:
A convolution combines pixels in a source image that you specify with neighboring pixels to produce an image. You can achieve a wide variety of imaging operations by using the convolution filter, which includes blurring, edge detection, sharpening, embossing, and beveling effects.

3.11.1Mean Filter: The idea of mean filtering is simply to replace each pixel value in an image with the mean (`average') value of its neighbours, including itself. This has the effect of eliminating pixel values which are unrepresentative of their surroundings. Mean filtering is usually thought of as a convolution filter. Like other convolutions it is based around a kernel, which represents the shape and size of the neighbourhood to be sampled when calculating the mean. Often a 3×3 square kernel is used, as shown in Figure 1, although larger kernels (e.g. 5×5 squares) can be used for more severe smoothing. (Note that a small kernel can be applied more than once in order to produce a similar - but not identical - effect as a single pass with a large kernel.)

3.11.2Blur Filter: This method makes the image blur. It uses this matrix:

1	2	3	2	1
2	4	5	4	2
3	5	6	5	3
2	4	5	4	2
1	2	3	2	1

3.11.3. Sharpen Filter: This method ensures the image to be sharpener. It uses this matrix:

0	-1	0
-1	5	-1
0	-1	0

3.11.4. Gaussian Filter: The Gaussian smoothing operator is a 2-D convolution operator that is used to `blur' images and remove detail and noise. In this sense it is similar to the mean filter, but it uses a different kernel that represents the shape of a Gaussian (`bell-shaped') hump.

$$G(x) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{x^2}{2\sigma^2}}$$

3.12. Two Sources Filters:

These filters ensure to work on two different images so you can merge, intersect, add, subtract, difference, and move towards them.

3.13. Edge Detection:

The goal of edge detection is to mark the points in a digital image at which the luminous intensity changes sharply. Sharp changes in image properties usually reflect important events and changes in properties of the world. These include discontinuities in depth, discontinuities in surface orientation, and changes in material properties and variations in scene illumination. Edge detection is a research field within image processing and computer vision, in particular within the area of feature extraction.

3.13.1. Homogeneity Edge Detection: If we are to perceive an edge in an image, it follows that there is a change in color between two objects, for an edge to be apparent. To put it another way, if we were to take a pixel and store as it's value the greatest difference between it's starting value and the values of it's eight neighbors, we would come up with black where the pixels are the same, and trend towards white the harder the color difference was. We would detect the edges in the image.

Furthermore, if we allowed a threshold to be set, and set values below this to 0, we could eliminate soft edges to whatever degree we desire. The code to do this is followed by an example at threshold 0 and one at threshold 127.

3.13.2. Difference Edge Detection: The difference edge detection works in a similar way, but it detects the difference between pairs of pixel around the pixel we are setting. It works out the highest value from the difference of the four pairs of pixels that can be used to form a line through the middle pixel. The threshold works the same as the homogeneity filter. Again, here is the code, followed by two examples, one with no threshold, and one with a threshold of 127.

3.13.3. Sobel Edge Detection: Based on this one-dimensional analysis, the theory can be carried over to two-dimensions as long as there is an accurate approximation to calculate the derivative of a two-dimensional image. The Sobel operator performs a 2-D spatial gradient measurement on an image. Typically it is used to find the approximate absolute gradient magnitude at each point in an input grayscale image. The Sobel edge detector uses a pair of 3x3 convolution masks, one estimating the gradient in the x-direction (columns) and the other estimating the gradient in the y-direction (rows). A convolution mask is usually much smaller than the actual image. As a result, the mask is slid over the image, manipulating a square of pixels at a time. The actual Sobel masks are shown below:

-1	0	+1
-2	0	+2
-1	0	+1

Gx

+1	+2	+1
0	0	0
-1	-2	-1

Gy

The magnitude of the gradient is then calculated using the formula:

$|G| = \sqrt{Gx^2 + Gy^2}$

An approximate magnitude can be calculated using:

$|G| = |Gx| + |Gy|$

3.13.4. Canny Edge Detection: This is a multi-step edge detection procedure by Canny. The purpose of the following two methods is to detect edges with noise suppressed at the same time.

1- Smooth the image with a Gaussian filter to reduce noise and unwanted details and textures.

$$g(m,n) = G_\sigma(m,n) * f(m,n)$$

Where $$G_\sigma = \frac{1}{\sqrt{2\pi\sigma^2}} e^{(-\frac{m^2+n^2}{2\sigma^2})}$$

2- Compute gradient of using any of the gradient operations (Roberts, Sobel, Prewitt, etc) to get:

$$M(n,n) = \sqrt{g^2{}_m(m,n) + g_n{}^2(m,n)}$$

and

$$\theta(m,n) = \arctan(\frac{g_n(m,n)}{g_m(m,n)})$$

3- Threshold M:

$$M_T(m,n) = \begin{cases} M(m,n) & \text{if } M(m,n) > T \\ 0 & \text{otherwise} \end{cases}$$

Where is so chosen that all edge elements are kept while most of the noise is suppressed.

4- Suppress non-maxima pixels in the edges in obtained above to thin the edge ridges (as the edges might have been broadened in step 1). To do so, check to see whether each non-zero $M_T(m,n)$ is greater than its two neighbors along the gradient direction $\theta(m,n)$. If so, keep $M_T(m,n)$ unchanged, otherwise, set it to 0.

5- Threshold the previous result by two different thresholds and (where) to obtain two binary images and. Note that compared to, has less noise and fewer false edges but larger gaps between edge segments.

6- Link edges segments in to form continuous edges. To do so, trace each segment in to its end and then search its neighbors in to find any edge segment in to bridge the gap until reaching another edge segment in.

3.14. Blob Counter:

The Blob Counter filter counts all blobs (objects or contours) within an image. The filter contains two filters for blobs. One filter excludes all blobs outside a minimum and maximum size. The second filter excludes all blobs whose seven "hu invariants" are outside a minimum and maximum value. Hu invariants are proved to be invariants to the image scale, rotation, and reflection except the seventh one, whose sign is changed by reflection.

3.15. Connected Component Labeling:

Connected component labeling works by scanning an image, pixel-by-pixel (from top to bottom and left to right) in order to identify connected pixel regions, i.e. regions of adjacent pixels which share the same set of intensity values V. (For a binary image V={1}; however, in a gray level image V will take on a range of values, for example: V={51, 52, 53, ..., 77, 78, 79, 80}.)
Connected component labeling works on binary or gray level images and different measures of connectivity are possible. However, for the following we assume binary input images and 8-connectivity. The connected components labeling operator scans the image by moving along a row until it comes to a point p (where p denotes the pixel to be labeled at any stage in the scanning process) for which V={1}. When this is true, it examines the four neighbors of p which have already been encountered in the scan (i.e. the neighbors (i) to the left of p, (ii) above it, and (iii and iv) the two upper diagonal terms). Based on this information, the labeling of p occurs as follows:
If all four neighbors are 0, assign a new label to p, else
If only one neighbor has V= {1}, assign its label to p, else
If one or more of the neighbors have V= {1}, assign one of the labels to p and make a note of the equivalences.
After completing the scan, the equivalent label pairs are sorted into equivalence classes and a unique label is assigned to each class. As a final step, a second scan is made through the image, during which each label is replaced by the label assigned to its equivalence classes. For display, the labels might be different gray levels or colors.

3.16. Simple Skeletonization:
Skeletonization is a process for reducing foreground regions in a binary image to a skeletal remnant that largely preserves the extent and connectivity of the original region while throwing away most of the original foreground pixels. The skeleton/MAT can be produced in two main ways. The first is to use some kind of morphological thinning that successively erodes away pixels from the boundary (while preserving the end points of line segments) until no more thinning is possible, at which point what is left approximates the skeleton. The alternative method is to first calculate the distance transform of the image. The skeleton then lies along the singularities (i.e. creases or curvature discontinuities) in the distance transform. This latter approach is more suited to calculating the MAT since the MAT is the same as the distance transform but with all points off the skeleton suppressed to zero.

3.17. Gamma Correction

To apply a gamma filter, each color is factored using the inverse power of the gamma value (Gamma is normally restricted to the range 1/5 to 5)

$$color' = 255 * (\frac{color}{255})^{\frac{1}{gamma}}$$

Since the gamma calculation involves powers, and for a given value of gamma and color the output color value is constant, it makes sense to pre-calculate the new color values and store them in an array so the calculation doesn't need to be repeated each time. Then the new value can be looked up simply as the index of the entry in a one-dimensional array. For each pixel in the image array, read the red, green and blue values and then replace them with their lookup within the gamma tables.

3.18. Conservative Smoothing

Like most noise filters, conservative smoothing operates on the assumption that noise has a high spatial frequency and, therefore, can be attenuated by a local operation which makes each pixel's intensity roughly consistent with those of its nearest neighbors. This is accomplished by a procedure which first finds the minimum and maximum intensity values of all the pixels within a windowed region around the pixel in question. If the intensity of the central pixel lies within the intensity range spread of its neighbors, it is passed onto the output image unchanged. However, if the central pixel intensity is greater than the maximum value, it is set equal to the maximum value; if the central pixel intensity is less than the minimum value, it is set equal to the minimum value.

3.19. Adaptive Smoothing

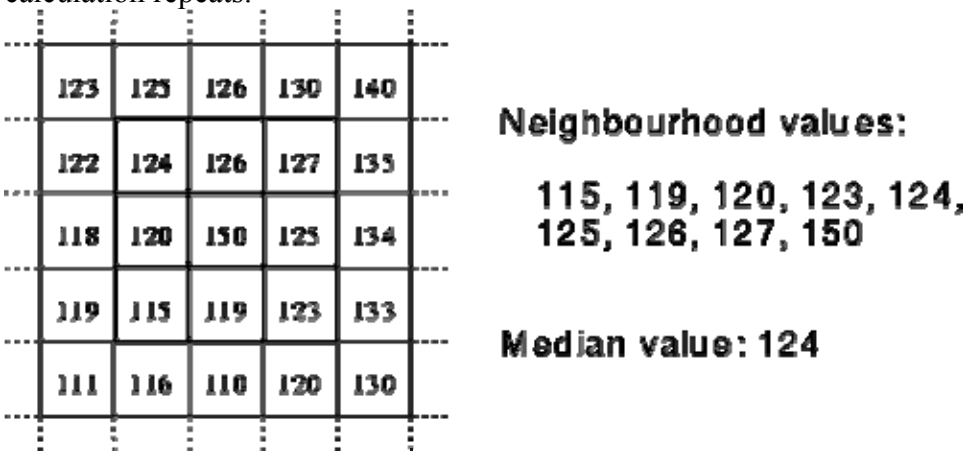
Adaptive Smoothing is a new image smoothing method developed by Image Fusion Systems Research. While traditional smoothing methods use the same window shape and size, or the same filter to smooth an image independent of its content, in adaptive smoothing, window shape and size are varied across the image domain depending on local image content. In adaptive smoothing a window is sized according to the gradient magnitude locally and shaped in such a way that it has a shorter side across edges compared to along edges. This mechanism maintains edge details while smoothing random noise.

Adaptive smoothing reduces noise similar to traditional smoothing but unlike traditional smoothing, it does not blur image edges

3.20 Median Filter:

The median filter is a non-linear digital filtering technique, often used to remove noise from images or other signals. The idea is to examine a sample of the input and decide if it is representative of the signal. This is performed using a window (n*n and n is an odd number)

The values in the window are sorted into numerical order; the median value is selected as the output. The oldest sample is discarded, a new sample acquired, and the calculation repeats.



3.21. Fourier transformation:

Fourier transformation allows us to handle the pictures in frequency domain instead of spatial domain. The simplest Fourier filters are low-pass and high-pass filters, which filter out high or low frequencies (rapid changes in pixel values or average values).

- Low Pass Filter: An image can be blurred by, for each pixel, calculating the weighed average of that pixel and its neighbors, using a convolution matrix. We can also blur an image, by multiplying the spectrum with the transfer function of a low pass filter. The transfer function of a low pass filter makes all high frequency components zero or at least weakens them, while keeping or amplifying the low frequency ones.
- High Pass Filter: high pass filter on an image keeps only the high frequency components. By applying a high pass filter, the DC component (the center of the spectrum) of the spectrum is removed too, the resulting image can have negative pixel values and it needs normalization.

4. SCREENSHOTS:

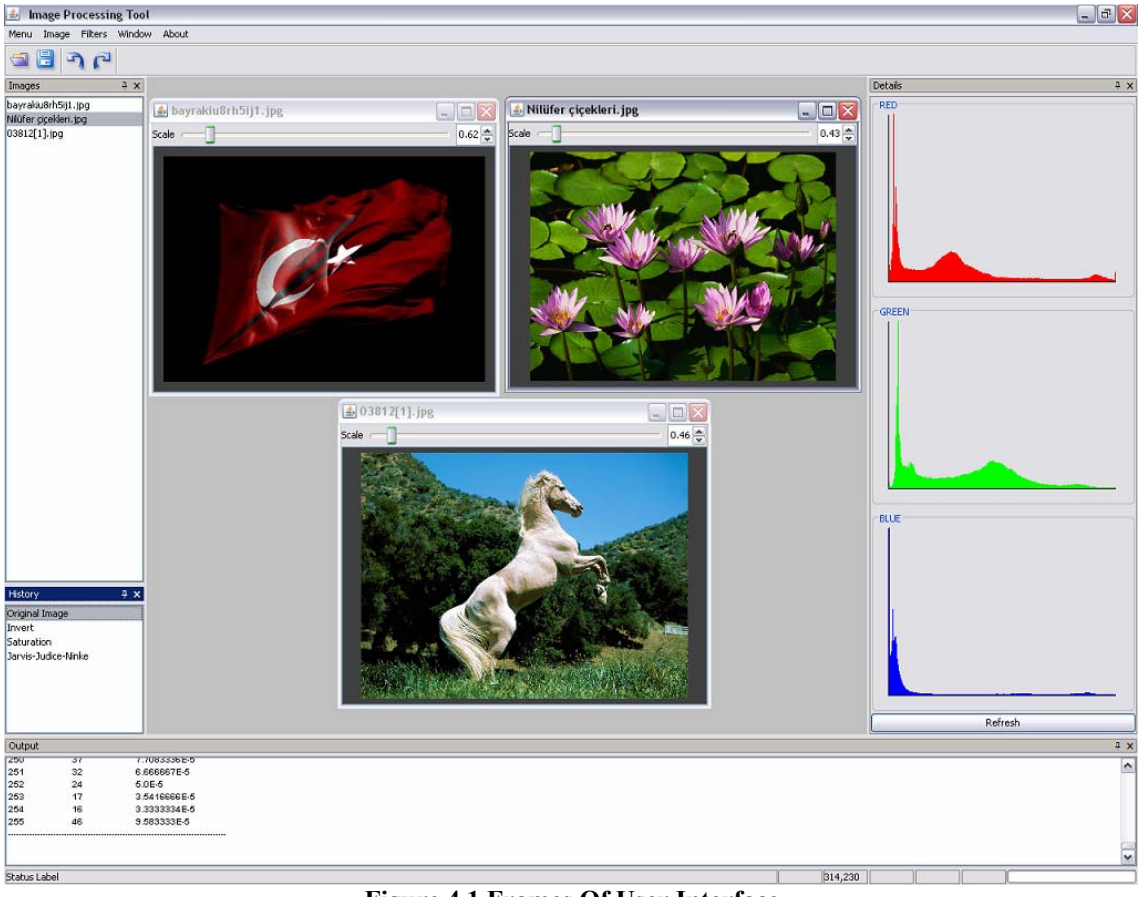


Figure 4.1 Frames Of User Interface

There are 5 frames in the graphic user interface. Their names are Images, History, Details, Output, and Main frame.

Images: In this frame the names of the pictures which are being shown in main frame are written. You can open many images and they are all shown in different frames.

History: In this frame the filters that have been applied to the image. When the selected image changes at the same time the history of the new image is written in History Frame.
Details: This frame shows the details about the image like histogram and other image statistics.
Output: This frame is giving any information to the user about the filter that is being applied. The warnings are shown in pop-up menus and the others are shown in Output frame:
Main Frame: This frame contains the images that are being shown. One or more image can be shown at the same time.

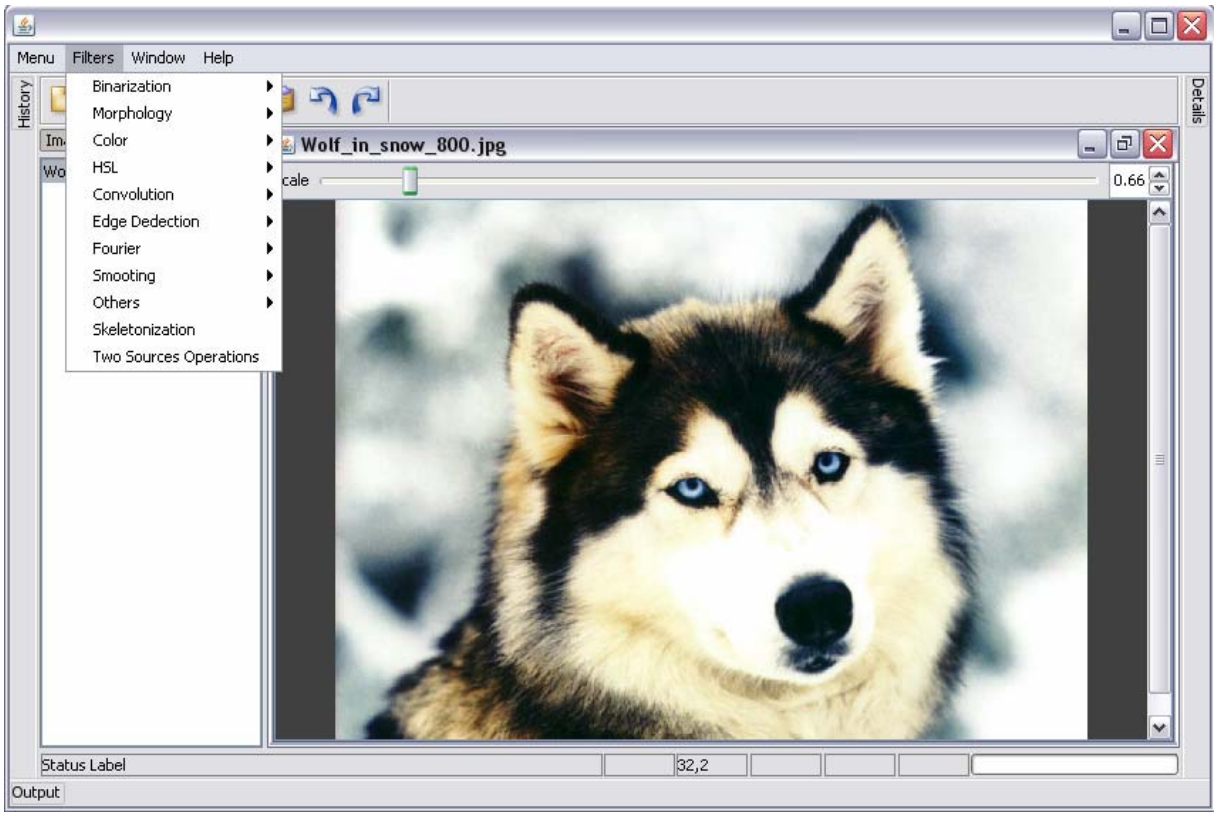


Figure 4.2 General User Interface

General use of application is very simple. Filters can be selected from the filters menu for applying to image. Filters have been classified in 11 main sections.

Figure 4.3 Threshold Binarization

Binarization section has 8 subsections: All these subsections create a binary image from the selected image. In here threshold binarization was applied to image.

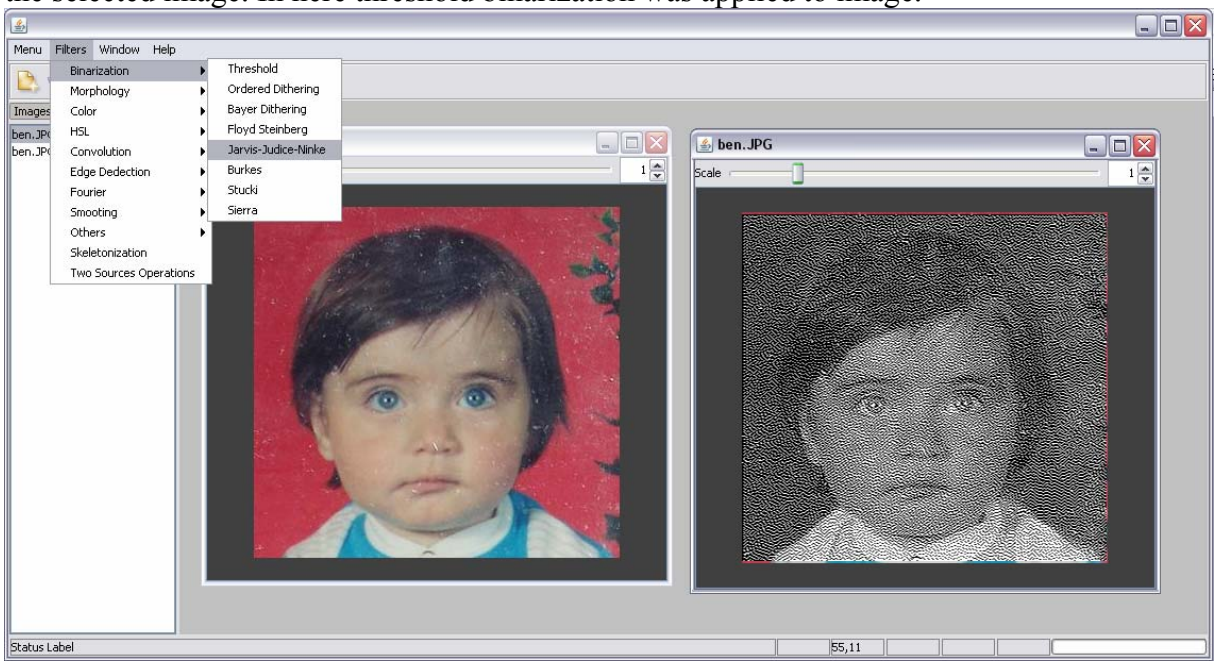


Figure 4.4 Jarvis-Nudice-Dunke Binarization

Jarvis-Nudice-Dunke Binarization was applied to image.

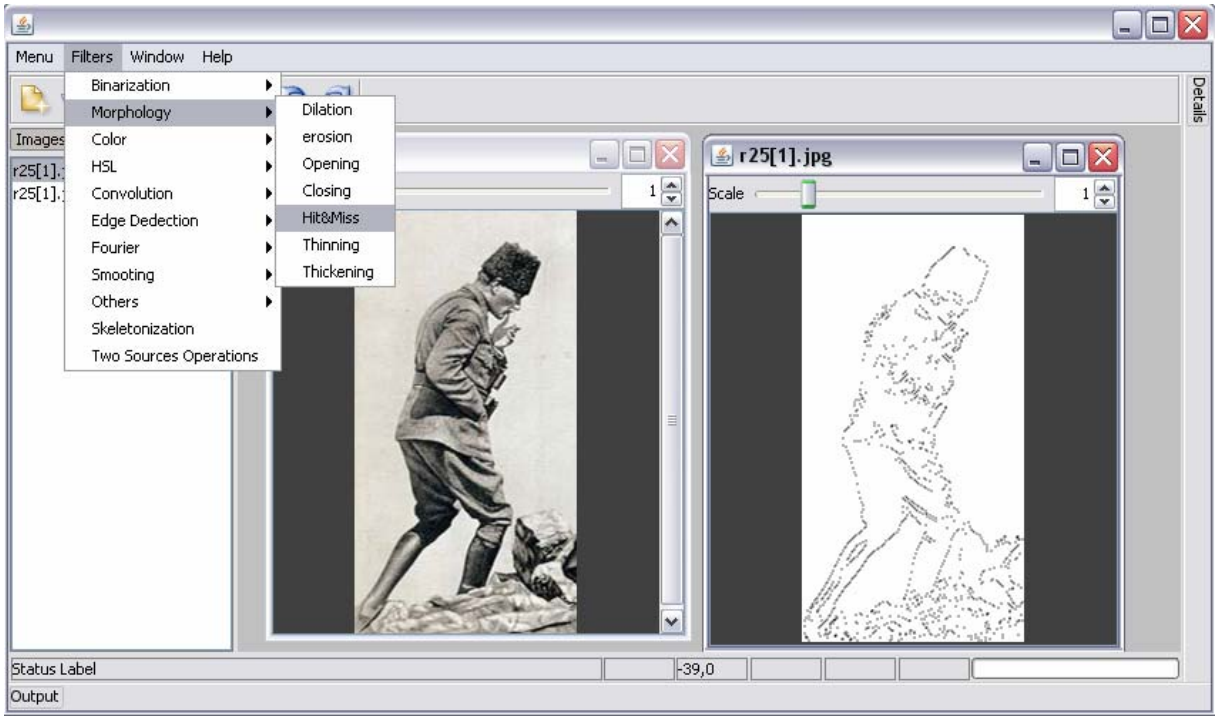


Figure 4.5 Hit&Miss Filter

Morphology section has 7 subsections that apply basic morphologic methods. For applying morphologic methods image must be binary. If image is not a binary image it is wanted to apply a binarization method from user. Hit&Miss method was applied to image in here.

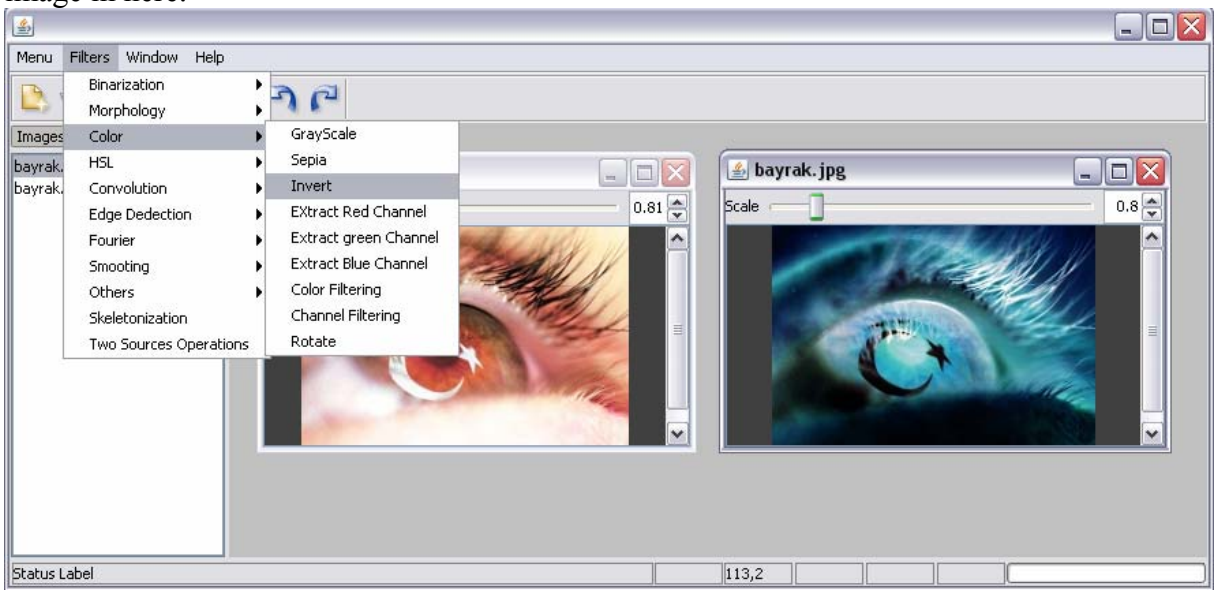


Figure 4.6 Invert

Color section has 9 subsections. These subsections are related with RGB color space. Invert method was applied to image.

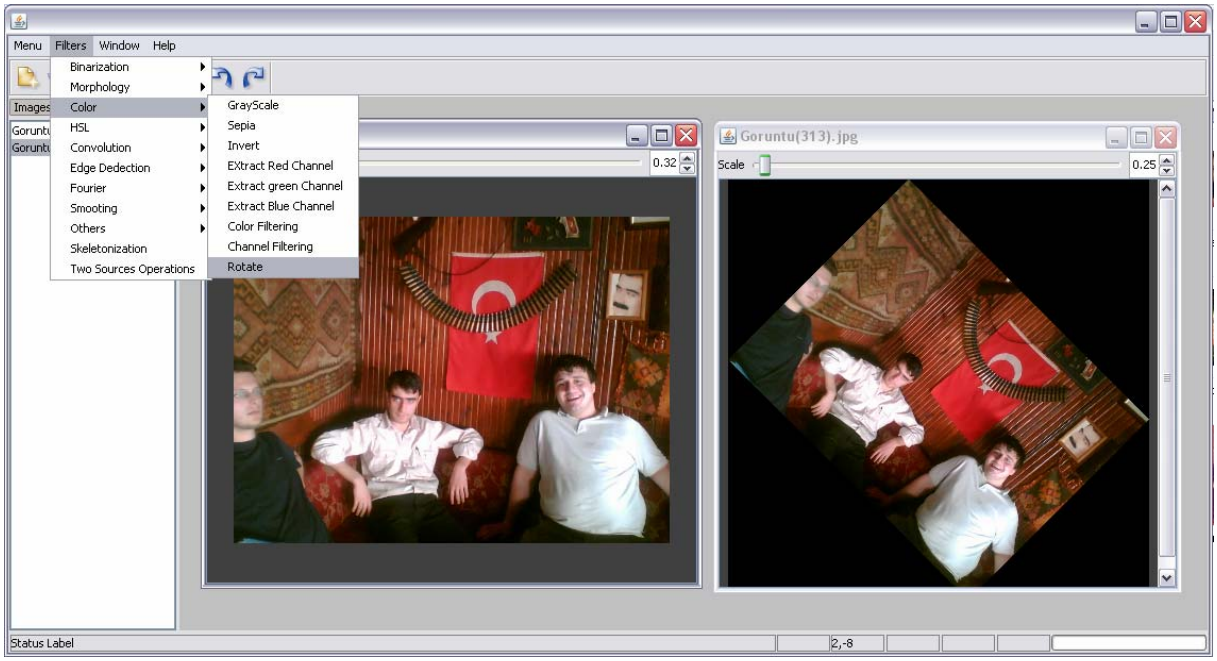


Figure 4.7 Rotate

The image was rotated 45 degree in here.

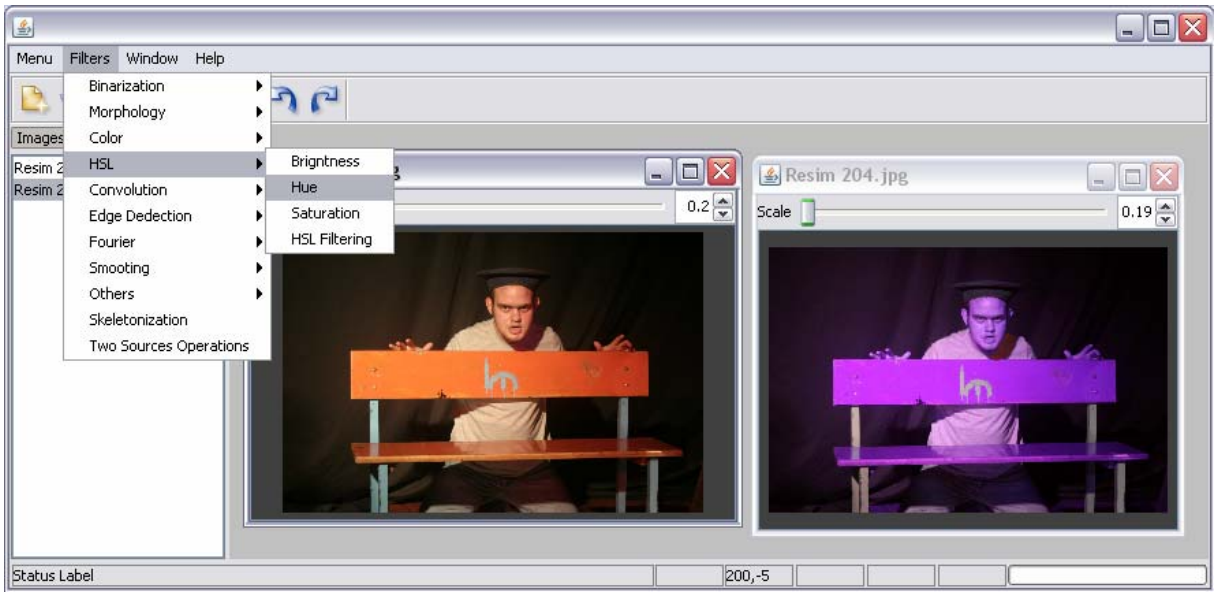


Figure 4.8 Hue Modifier

HSL section has 4 subsections. These subsections are related with HSB color space. Hue value was changed in here.

Figure 4.9 Saturation Modifier

Saturation of the image was changed in here.

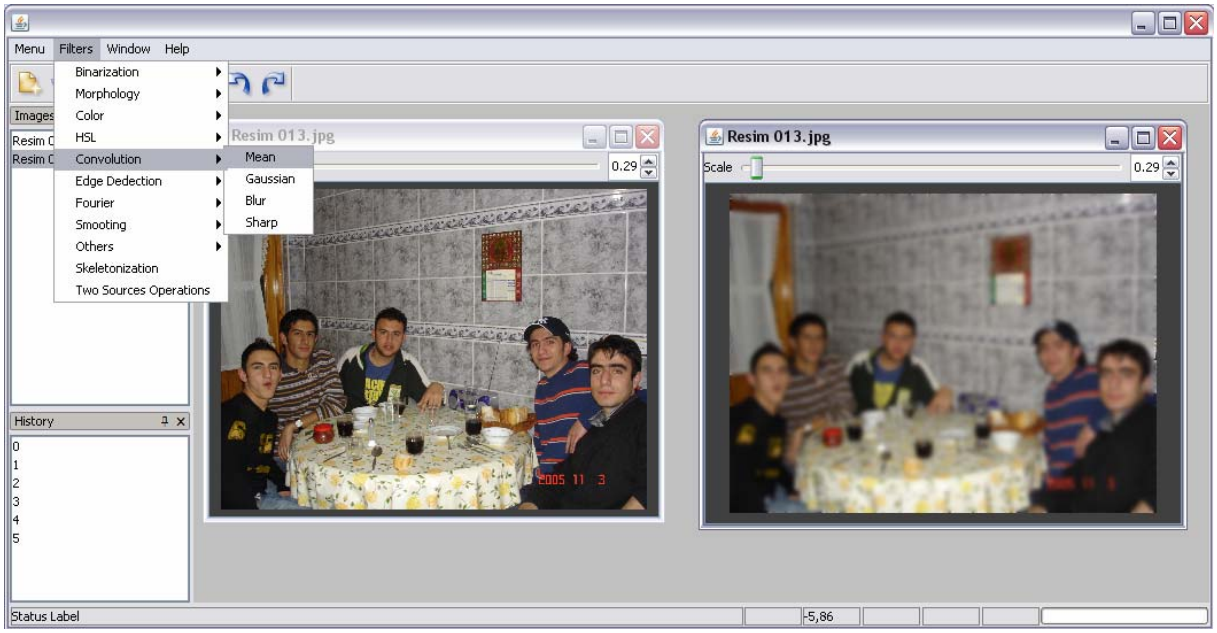


Figure 4.10 Mean Filter

Convolution section has 4 subsections. These convolution methods are applied with the matrix which is taken from the user. In here the effect of Mean Filter on the image is seen.

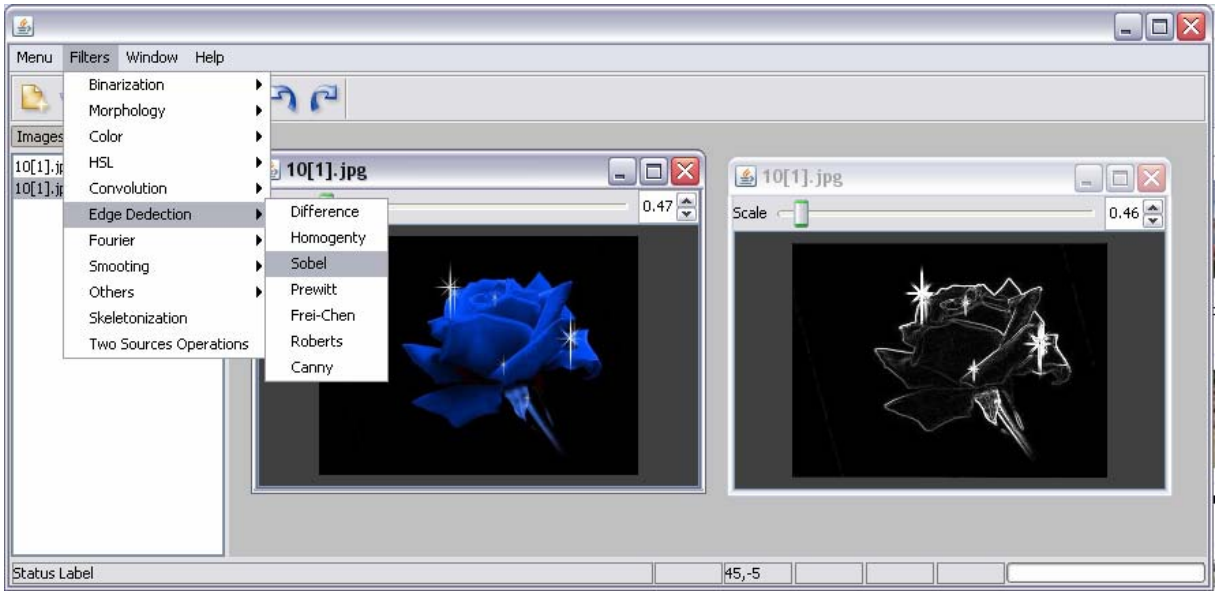


Figure 4.11 Sobel Filter

Edge detection section has 7 subsections. These sections are the most popular edge detection methods. Here the edges were detected with Sobel Filter.

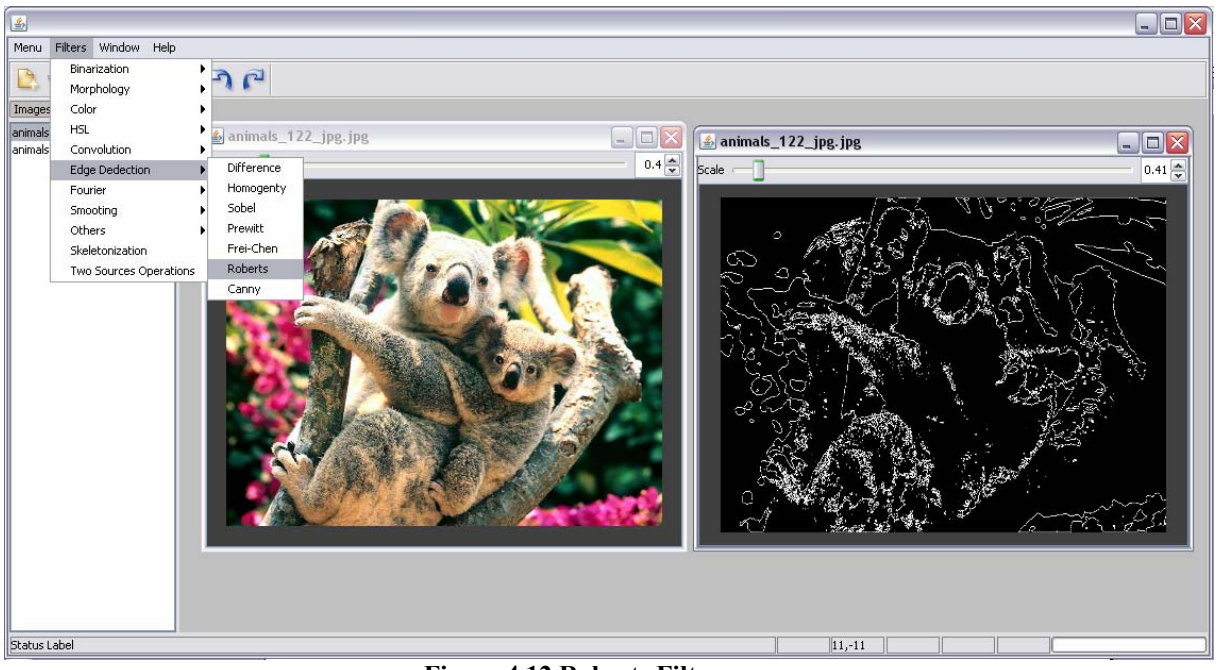


Figure 4.12 Roberts Filter

Another Edge Detection method (Robert Edge Detection) was applied to the image. The more image gets complicated detection of the edges become harder.

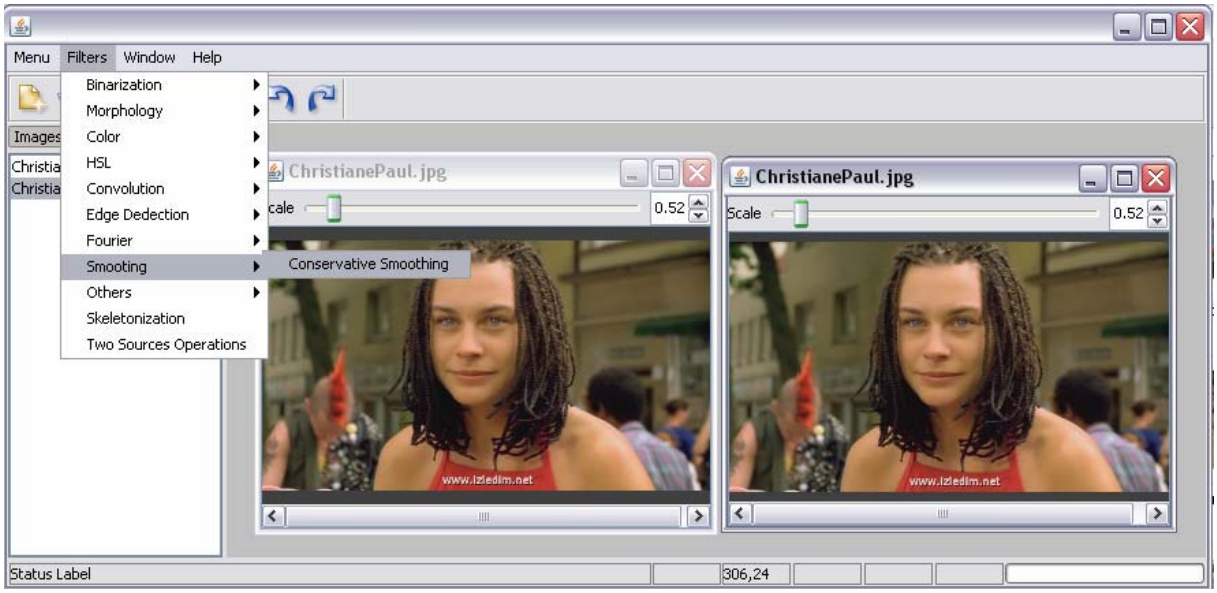


Figure 4.13 Conservative Smoothing

Smoothing section has only one subsection: Conservative smoothing.

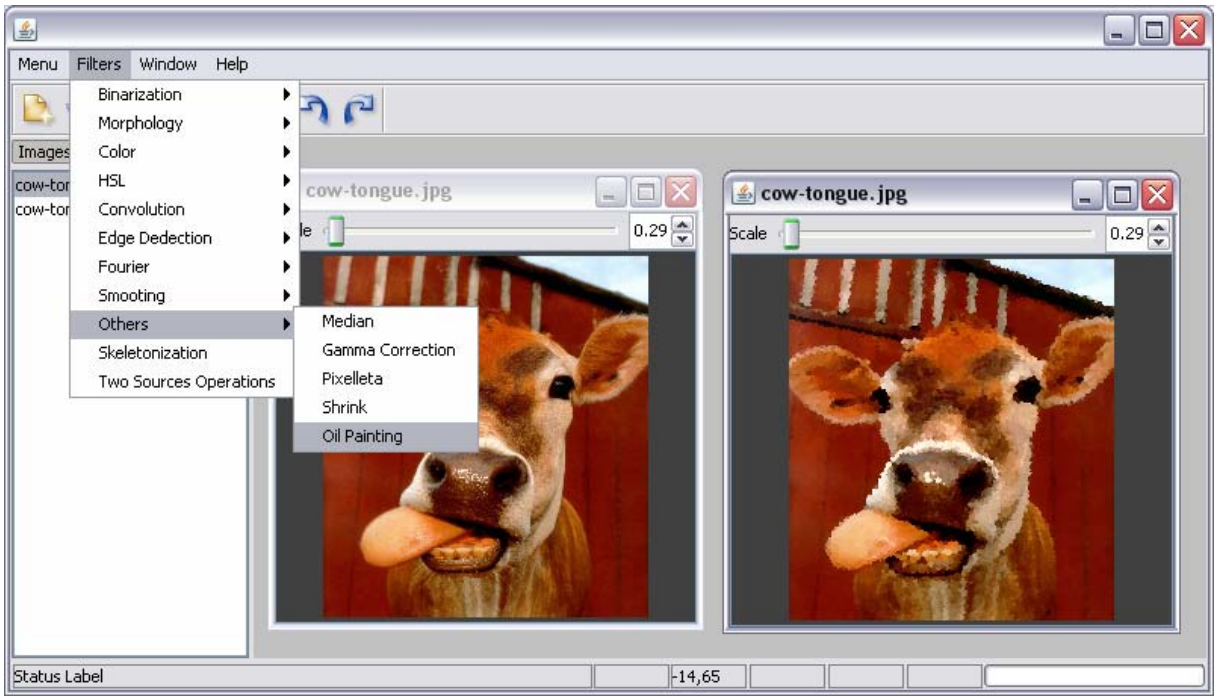


Figure 4.14 Oil Painting

Others section has 5 subsections. These subsections are generally known methods but are suitable to take place in any classification. So they were collected in Others section. The effect of Oil Painting Filter is seen in here.

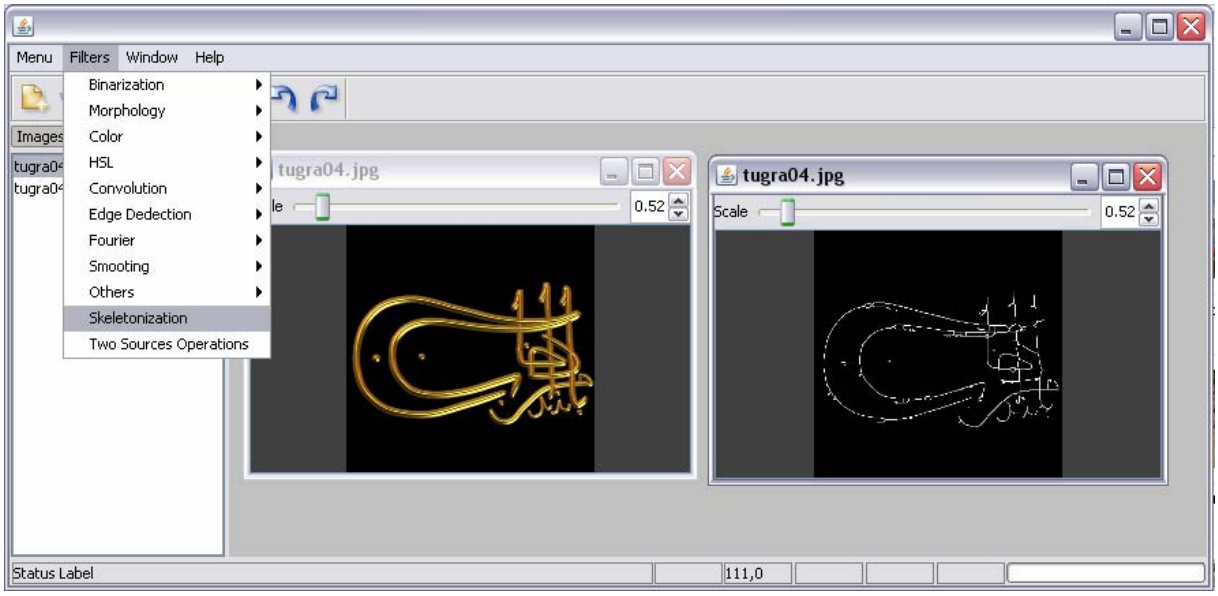


Figure 4.15 Skeletonization

Skeletonization section has no subsection. Simple skeletorization method is applied to the image. If the image is not a binary image it is converted to binary image then skeletonization method is applied.

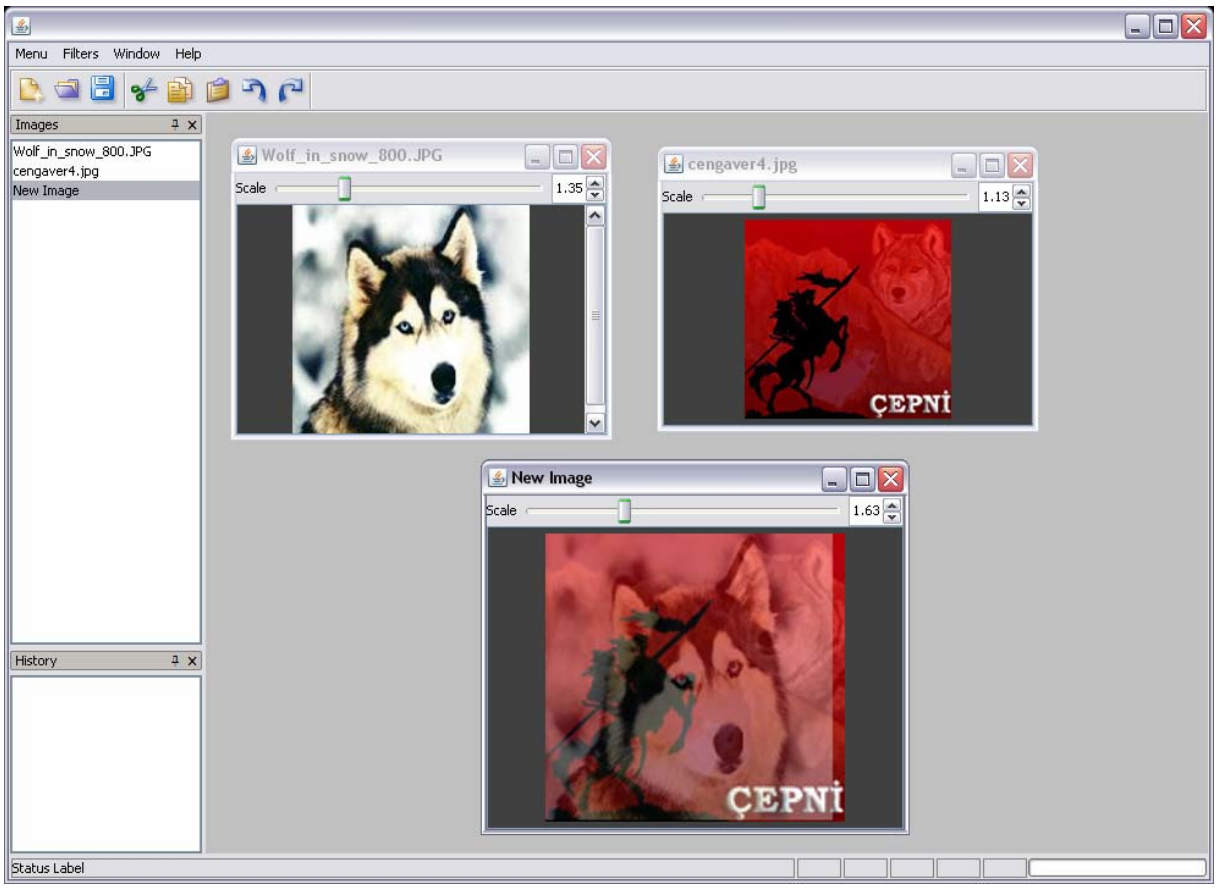


Figure 4.16 Merge

Two Sources Operations has three subsections: Add, Subtract, and Merge. These methods are applied with two images. When Two Sources Operations section is selected a new window appears which includes the names of the current images. User can select the second image from this window and apply the selected method. In here the new image is the merge of the other two images.

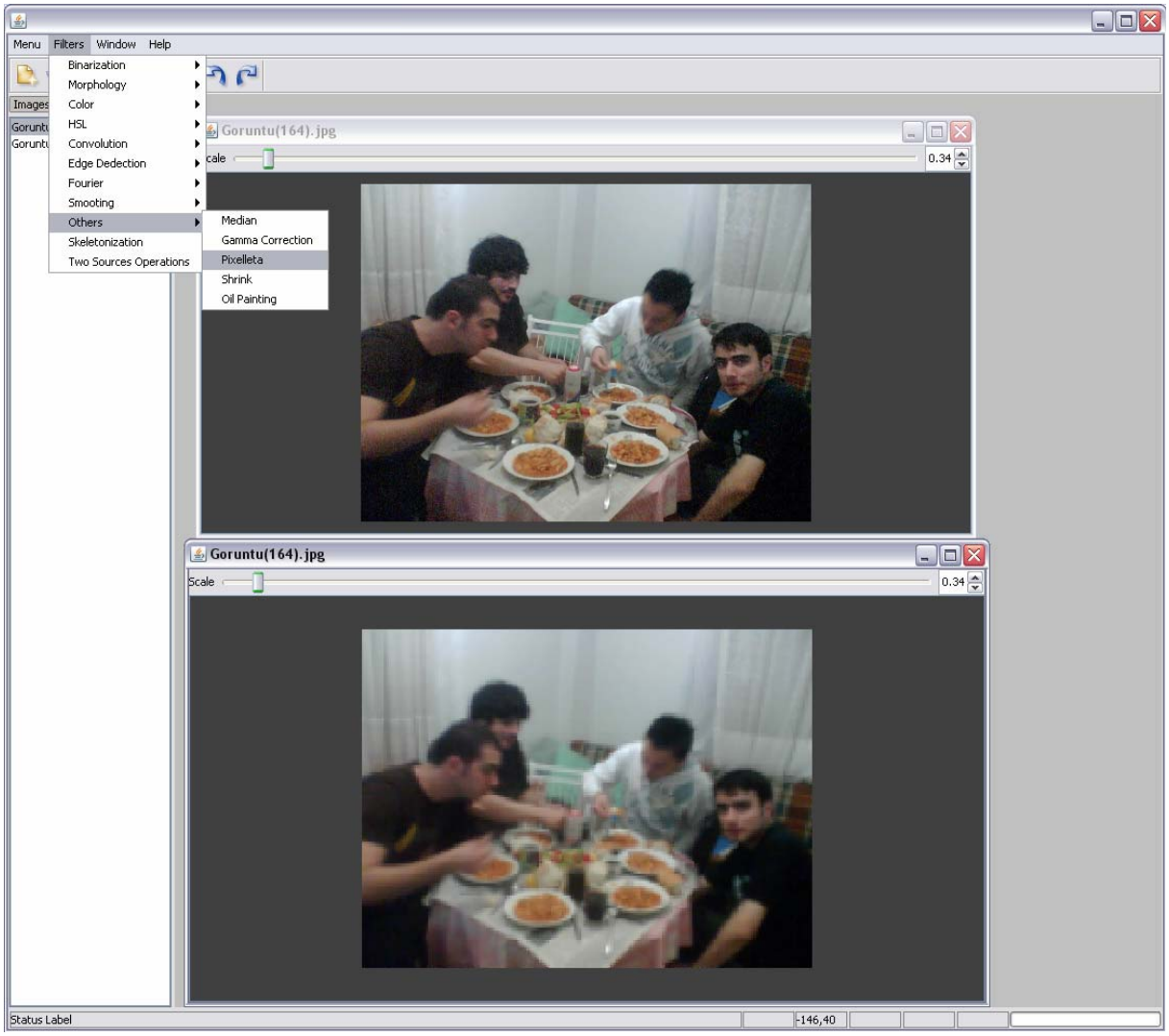


Figure 4.17 Pixelleta

The effect of Pixelleta filters is seen here.

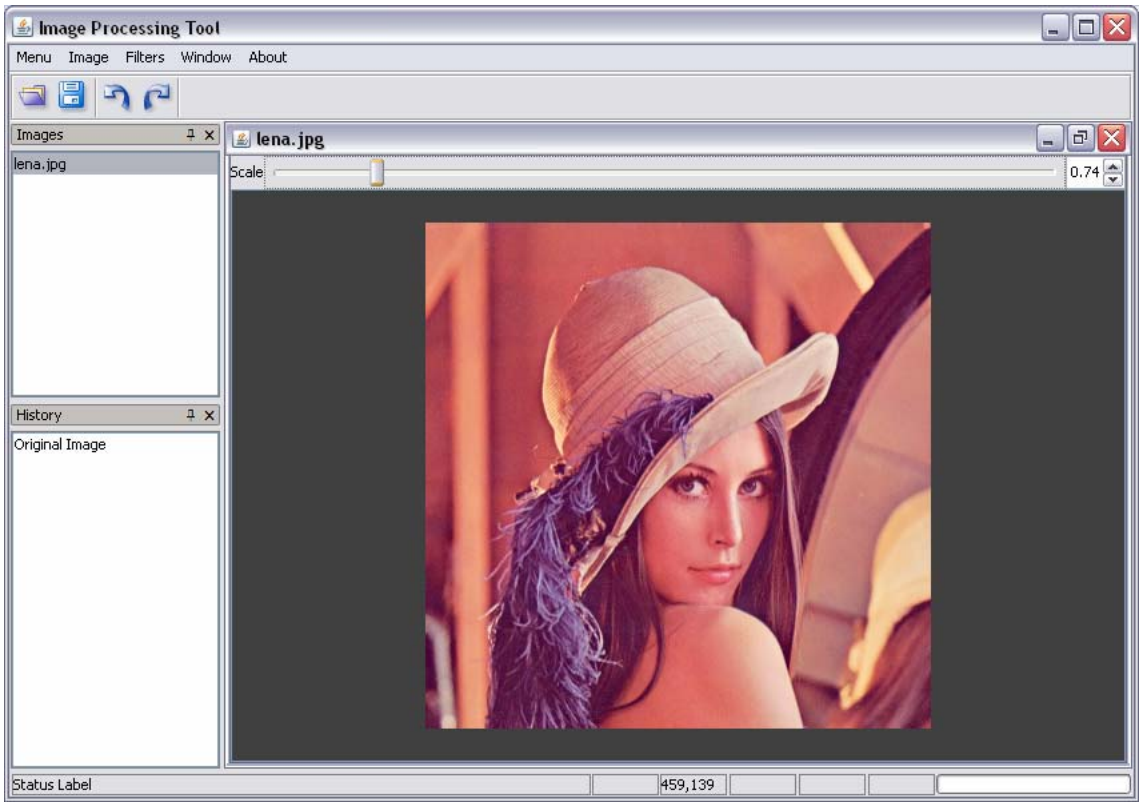


Figure 4.18 Original Lena

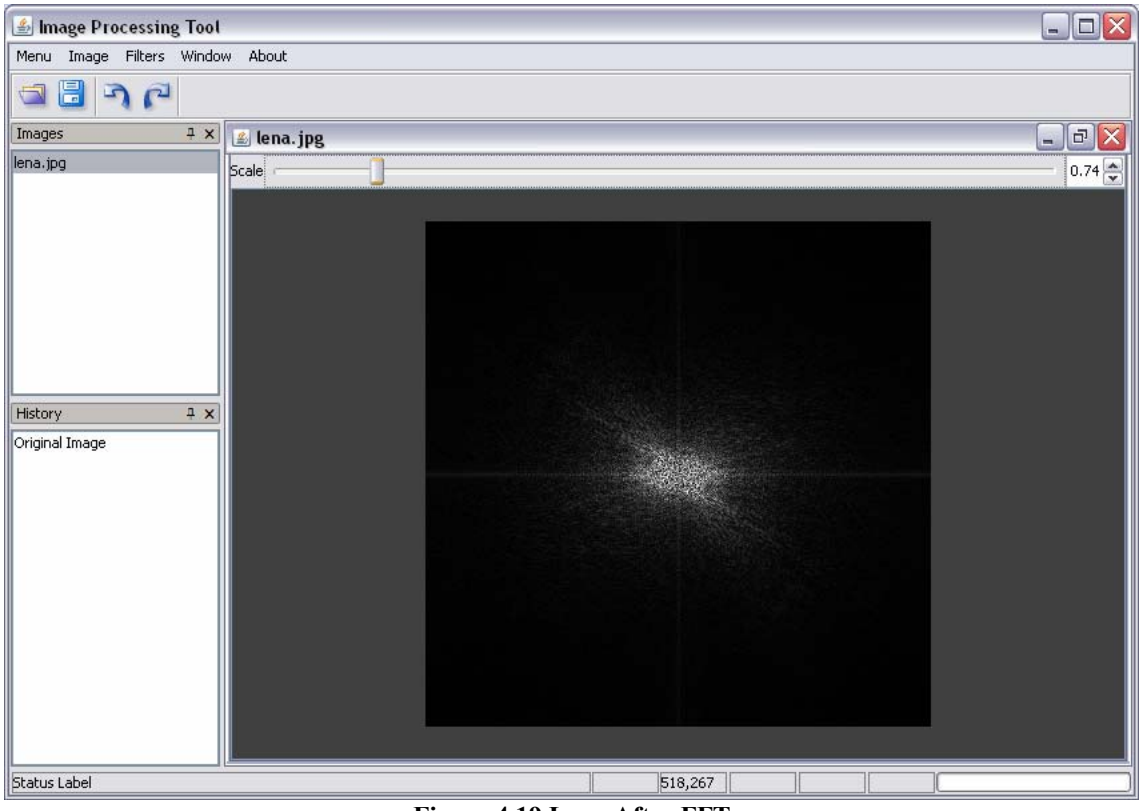


Figure 4.19 Lena After FFT

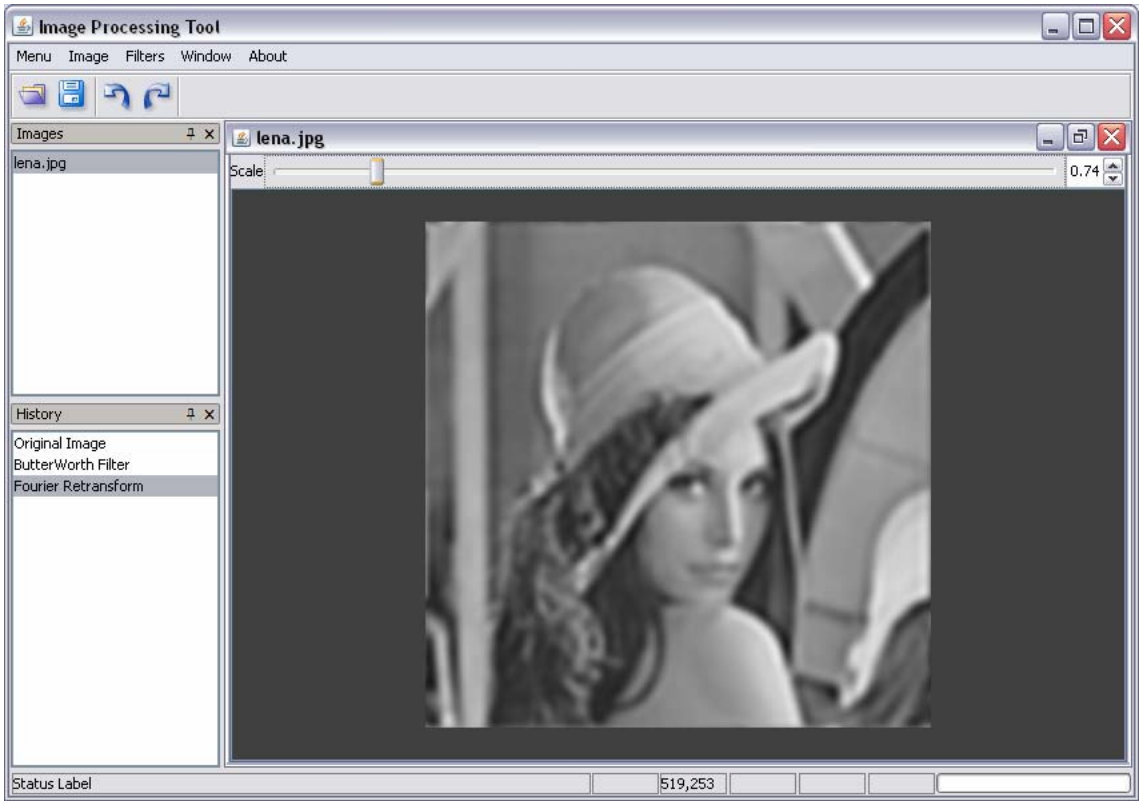


Figure 4.20 Retransform After Butterworth Filter

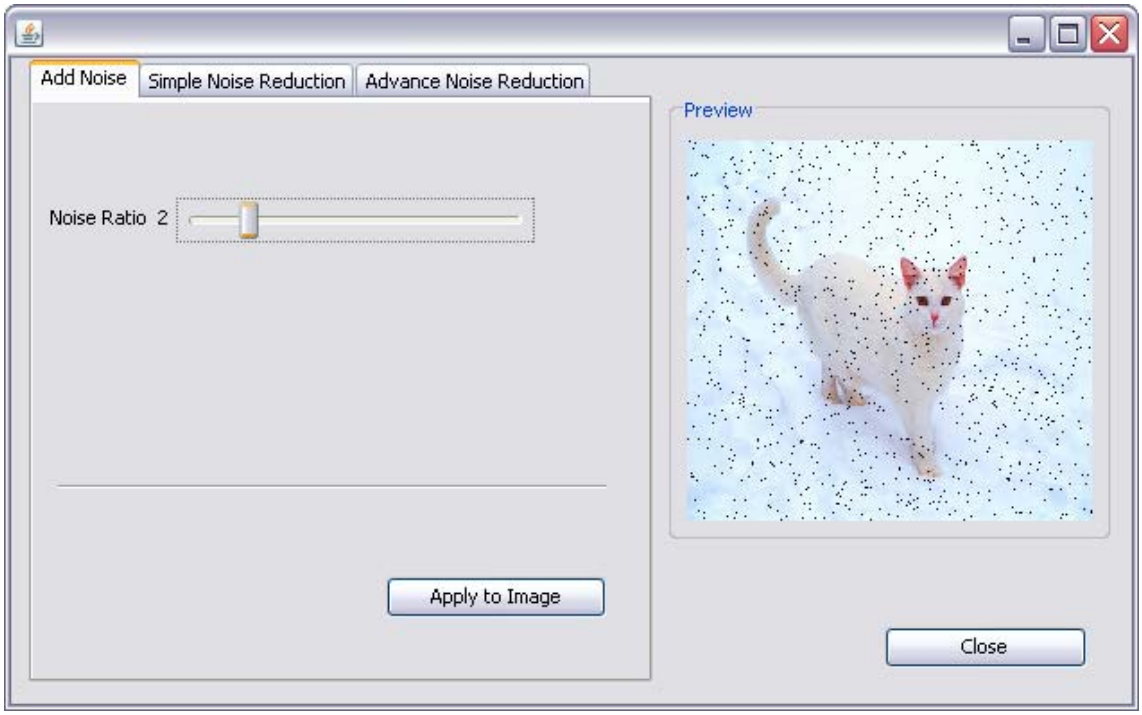


Figure 4.21 Adding Noise

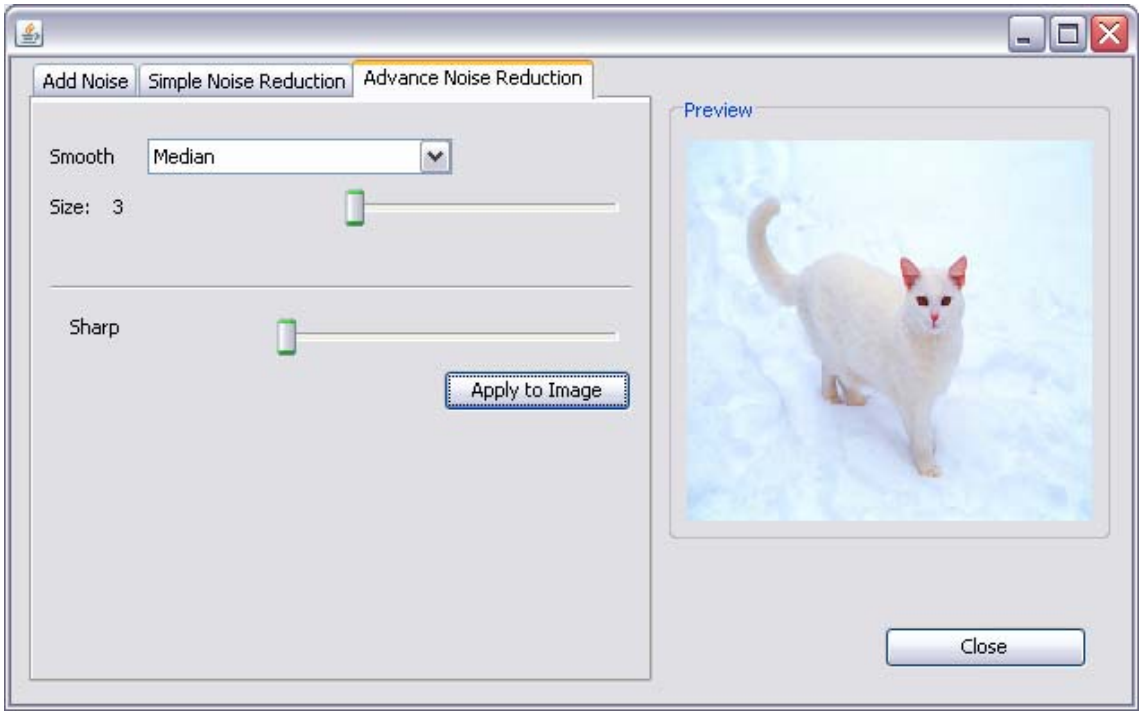


Figure 4.22 Noise Reduction

CONCLUSION:

Thinking the project we realized, we have developed and designed an image processing tool. The processes handled by this application can be increased. Besides, the commonly used processes are included in the application, in fact. When comparing this application with its equivalents, it has been seen that the application functions with higher effectiveness. Moreover, the functioning parameters are left for the users as far as possible.

REFERENCES:

[1] HyperMedia Image Proceing,http://www.cee.hw.ac.uk/hipr/html/hipr_top.html
[2] Image Processing Fundamentals,
http://www.ph.tn.tudelft.nl/Courses/FIP/noframes/fip-Morpholo.html
[3] Gonzalez & Woods, Digital Image Processing, 2nd Edition, Prentice Hall, 2002
[4] Russ, John C.(1995), THE IMAGE PROCESSING HANDBOOK, 2nd edition, CRC Press,1995
[5] Crane, Randy, A. Simplified Approach to Image Processing: Classical and Modern Techniques in C, Prentice Hall, 1996

ÖZGEÇMİŞ:

Adı Soyadı	Akın MOĞULKOÇ
Doğum Tarihi	20.05.1985
Doğum Yeri	Sivas/Kangal
Lise	Kayseri Fen Lisesi
Staj Yaptığı Yerler.	Link Holding(Yazılım)
	Türk Telekom(Donanım)

ÖZGEÇMİŞ:

Adı Soyadı	Orçun ATAY
Doğum Tarihi	13.04.1985
Doğum Yeri	Zonguldak
Lise	Zonguldak Atatürk Anadolu Lisesi
Staj Yaptığı Yerler	Link Holding(Yazılım)
	Art Elektronik Ltd.(Donanım)