Photo Crafter
Image Processing and Manipulation

Rohita Darga (201201104)
Amar Budhiraja (201303009)
rohita.darga@students.iiit.ac.in
amar.budhiraja@students.iiit.ac.in

April 17, 2014

1 Abstract

Photo Crafter is an image processing application for Windows/Linux created using matlab that would allow for basic photo manipulations. Its GUI lets even the most inexperienced of users implement changes to their favorite images that will not only accent them, but will make them look like professional edits. This tool lets users add effects to their images making them the classic gray-scale, black and white, sepia or others. However, they are not limited to these effects. Additional filters such as Gaussian blur, edge detection and shadow focus are also available. Finally, the image can be compressed on demand as well. This tool helps amateur photographers dabble in image editing while saving space in their computer memory.

2 Introduction

Image processing has become one of the foremost runners in the software race. Almost every company has some use for it. For example, Facebook now implements image processing in the form of face detection. It determines the most prominent features in a person's face and matches it in other pictures in order to ease the process of tagging photos in the Facebook photo albums. Google now has the ability to search the internet using images instead of key words. It matches the images on the internet to the one that it is provided. All of these image processing techniques are profound and make a difference in the way technology is used today. It can effect the way criminals are caught, it can help detect diseases using images and etc.These techniques all start at a basic idea: interpreting the pixels. Image processing in the field today is attempting to mimic the human thought process in identifying different features in the images. This way we can remove human error in the way image recognition, and also make it a much quicker process. What takes human eyes hours to look through can now be processed in seconds. Photo Crafter just touches the basics with image processing. It is a beginners tool for amateur photographers to implement basic filters to their images with effects found on popular social medias. It dabbles in the infinite potential of image processing by providing common effects and filters. These effects are often the basis for greater image processing.

3 Procedure

3.1 Matlab

Matlab was a monumental tool in the creation of this application. Its raw computing power allowed for easy manipulation of the images and gave proper control of the tools. Matlab is the back-end processor for all the image manipulations in the application. It allowed for fast and easy matrix modifications and computations such as convolutions. Matlab also allowed for the creation of a simple but effective UI that showed the image progression as it was being edited. This lets the user understand the effects the editing is having on the image on a small but effective scale. The Mat-

lab Gui was highly customizable and therefore it made it very easy to create a simple but effective GUI.

3.2 Effects

The effects used in this application were created using a program that looked through all the pixels in the image and reduced or increased the corresponding color values. Take sepia for example, in this case the program would reduce the blues and greens and increase the reds in the image and so on for the other effects. Through this process, a blue effect, a gray-scale effect, sepia effect and many more were implemented. Let's have a walk through into the program. This program reads the image into the memory and reads the RGB values for every pixel. Then, new RGB values are created by modifying the existing values in order to keep the integrity of the image, but also changing the coloring scheme of the image. This is how the various effects are created. By different adjustments in the color values, various colors can be accented, highlighted or reduced.

```
Data = imread(Filename);
for i = 1 : r
    for j = 1: c
        R = RGB(i,j,1);
        G = RGB(i,j,2);
        B = RGB(i,j,3);
        r = 0.9*R + 0.1*G + 0.4*B;
        g = 0.3*R + 0.2*G + 0.4*B;
        b = 0.3*R + 0.2*G + 0.1*B;
        Data(i,j,1) = r ;
        Data(i,j,2) = g ;
        Data(i,j,3) = b ;
    end
end
```

3.3 Smoothening

These effects are created using simple Matlab filter functions. These are inbuilt into Matlab and require no extensive code. These filters when implemented from scratch are created using convolution functions. For example, the Gaussian blur is created using an impulse that is a Gaussian function. This is similar to the other filters. The average filter is a convolution function with an array that has a constant value of one divided by the size of the array for all of its values. This will reduce the noise in the image and blur it slightly as the pixel being affected is being averaged by the pixels around it. The disk filter is similar to the average function but instead, of just taking a square matrix of values to average, the disk filter averages the values within a circular radius of the pixel.

An example of both the inbuilt function and the psuedo algorithm is shown below.

```
my_image = imread(filenane);
e = imfilter(my_image,fspecial('average'));

#####  Average Filter  #######
C = imread('images_noise1.jpg');
m=[1 , 1 , 1, 1 ; 1 , 1 , 1 , 1 ; 1 , 1 , 1 , 1]
m = m/12;
E = conv2(C,m);
```

3.4 Edge Detection

Edge Detection can be created using simple Matlab filter functions as well. These filters can be created using different methods of convolution. The Prewitt filter is often used to aid in edge detection. In the filter, at each point of the image, the value becomes either the corresponding gradient vector or the norm of this vector. This is achieved using convolution once again. The Laplacian filter can be used to aid edge detection as well. Here the impulse function for the convolution is the Laplacian function. An example of both the inbuilt function and the psuedo algorithm is shown below.

The Prewitt filter nor the Laplacian filter created a perfect edge detect, therefore we attempted to create a seperate edge detection matrix in order to improve the results. In this method, we converted the image to black and white in order to make the edges more distinct.

```
my_image = imread(filenane);
e = imfilter(my_image,fspecial('prewitt'));

#####  Prewitt Filter  #####
C = imread('images_noise1.jpg');
m=[-1, 0 , 1 ; -1 , 0 , 1 ; -1 , 0 , 1];
```

```
E = conv2(C,m);

#####  Edge Detect    ######
sEdge = [-1,0,1;-2,0,2;-1,0,1];
C = imread(Filename);
F = im2bw(C, .65);
E = conv2(double(F), sEdge);
F = im2bw(E, .9);
```

3.5 Focus

Focusing is the idea where one part of the image is kept undisturbed while all the other regions of the image are progressively blurred relative to its distance from the undisturbed part of the image. There is another type of focusing where instead of blurring the image, the image is darkened. We implemented these two varieties of focusing in the application. The program works by first asking the user to select parts of the image they want in focus. These values are used in order to create a circle of "focus". Then, a Gaussian blur is implemented with a degree that is proportional to the distance from the circle of focus. This is similar to the shadow focus. In the shadow focus, the colors are multiplied by a value that is inversely proportionate to their distance from the circle. The code for shadow focus can be seen below. In the code, the distance is being calculated from the center of focus (Ln) and then compared to the distance to the edge of the image(L). Then the value of f is calculated and new rgb values are caculated using the product of the old values multiplied by f.

```
for i = 1:r
  for j = 1:c
    Ln = sqrt((r0-i)^2 + (c0-j)^2);
    f = (L-Ln)./L;
    if(f>1)
      f=1;
    end
    IMG(i,j,:) = f*img(i,j,:);
  end
```

3.6 Fisheye

The Fisheye effect is often used to replicate the fisheye lens found in cameras. The Fisheye lens creates a bulbous look for the image being taken by making it look convex in nature. The name of the lens came to be because the lens is based on how a fish would see in a wide hemispherical view under the water. This lens effect can be replicated using digital image processing. This effect is replicated in the application by initially padding the image in order to make it circular in nature. Then, a modification matrix is made to convert the image into the convex format. The below code shows the creation of the modification matrix. First the image size, origin, scale and exponent are taken. The exponent dictates how "convex" the image will be. Then the conversion matrix is built using the distance from the origin and the angle values in order to convert the image.

```
function U = fisheye_inverse(X,T)

  imageSize = T.tdata(1:2);
  exponent = T.tdata(3);
  origin = (imageSize+1)./2;
  scale = imageSize./2;

  x = (X(:,1)-origin(1))/scale(1);
  y = (X(:,2)-origin(2))/scale(2);
  R = sqrt(x.^2+y.^2);
  theta = atan2(y,x);

  cornerScale = min(abs(1./sin(theta)),abs(1./co
  cornerScale(R < 1) = 1;
  R = cornerScale.*R.^exponent;

  x = scale(1).*R.*cos(theta)+origin(1);
  y = scale(2).*R.*sin(theta)+origin(2);
  U = [x y];

end
```

3.7 Compression

Compression is the processing of converting a file into a file that uses less data to store its information. The idea is to save as much data as possible without losing the integrity of the image or file. This application uses discrete cosine transform in order to achieve this effect. However, the more the image is compressed, the image suffers in quality. The following code shows the compression process if the default matlab function is used.

```
for i=1:size(o, 1)% all rows
 rowDCT = dct2(double(o(i,:,k)));
 ci3(i,:,k)=idct(rowDCT(1:samplesThree), w)
end
```

4 Dataset

Type	Grayscale	RGB
jpg/jpeg	yes	yes
tiff	yes	yes
png	yes	yes
bitmap	yes	yes

5 Results

The results of this appliaction is more or less the scope of the application. The application first tested the smoothening effects of the Gaussian blur with different standard deviations. The three standard deviations tested were, one, three and ten. The following images show the changes the sigma value has on the original image().

Figure 2: Original Image

After, the Gaussian Blur was implemented with different standard deviations. We compared the different smoothening methods. These would include, the average filter, the Gaussian filter and the disk filter. The following figures show the effects of all three filters and a zoomed in section in order to compare the filters. The smoothening filter values are the following: Average[11,11] vs Gaussian(Sigma = 3) vs Disk(Radius = 20).

Figure 3: Gaussian Image w/ Sigma=1

Figure 4: Gaussian Image w/ Sigma=3

Figure 5: Gaussian Image w/ Sigma=10

As can be seen, the Average filter reduces the fine image details more than the other filters, and the Gaussian preserves the most details. Now, it is up to the user to determine which of the smoothening filters they would prefer.

Figure 6: Smoothing with Average

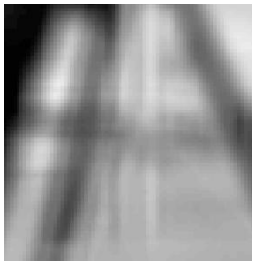

Figure 7: Close-up of Avgerage

From Figure 11 to Figure 14, the original image Lena is put through the different edge detection transformations.

Figures 15 to 17 are put through the different effects: grayscale, sepia and pink.

Finally, Figures 18 and 19 showcase the two different focus methods: Shadow Focus and Soft Focus.

Figure 8: Smoothing with Disk

Figure 9: Close-up of Disk

Figure 10: Smoothing with Gaussian

6 Summary

Using Matlab, which is a numerical computing environment, the images were sent through various forms of convolutions, color conversions, and matrices modification in order to create the various effects and effects. The user of the application can dictate the transforms they would want to happen to the image and when the button is pressed, matlab would appropriately convolute the image for the desired effect. The final image is then saved into the desired folder for the user so that they can access it in the future. This

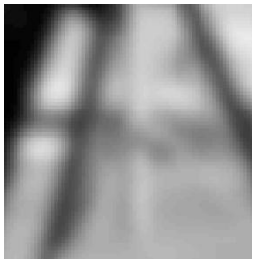

Figure 11: Close-up of Gaussian

Figure 12: Original

Figure 13: Laplacian filter

Figure 14: Prewitt Filter

Figure 15: Grayscale

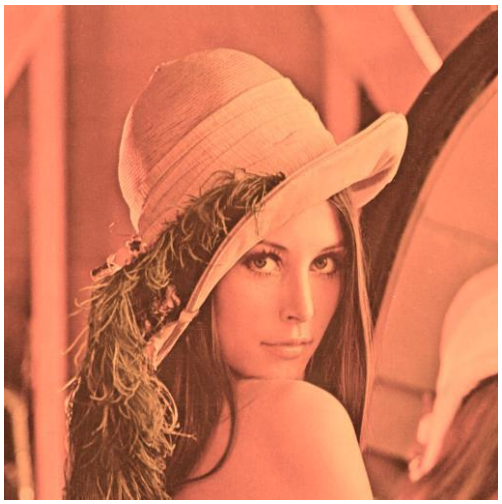

Figure 16: Pink Effect

way the user can keep all their edits without losing their handiwork.

Figure 17: Sephia Effect

Figure 18: Shadow Focus

Figure 19: Soft Focus

7 References

"Anyone Know the Formula to Get Sepia for an Image - Programming (C, C++, JAVA, VB, .NET Etc.)." Neowin.net. N.p., n.d. Web. 17 Apr. 2014.

"How Can I Implement This Visual Effect in MATLAB?" Image Processing. N.p., n.d. Web. 17 Apr. 2014.

"How to Make a Seamless Barrel (fisheye Lens) or Pincushion Distortion" - MATLAB Answers. N.p., n.d. Web. 17 Apr. 2014.

"Matlab Figure Focus - Old (highly Annoying) Changes." - MATLAB Answers. N.p., n.d. Web. 17 Apr. 2014.

"Thread Subject: Fisheye View in Matlab." Fisheye View in Matlab. N.p., n.d. Web. 17 Apr. 2014.

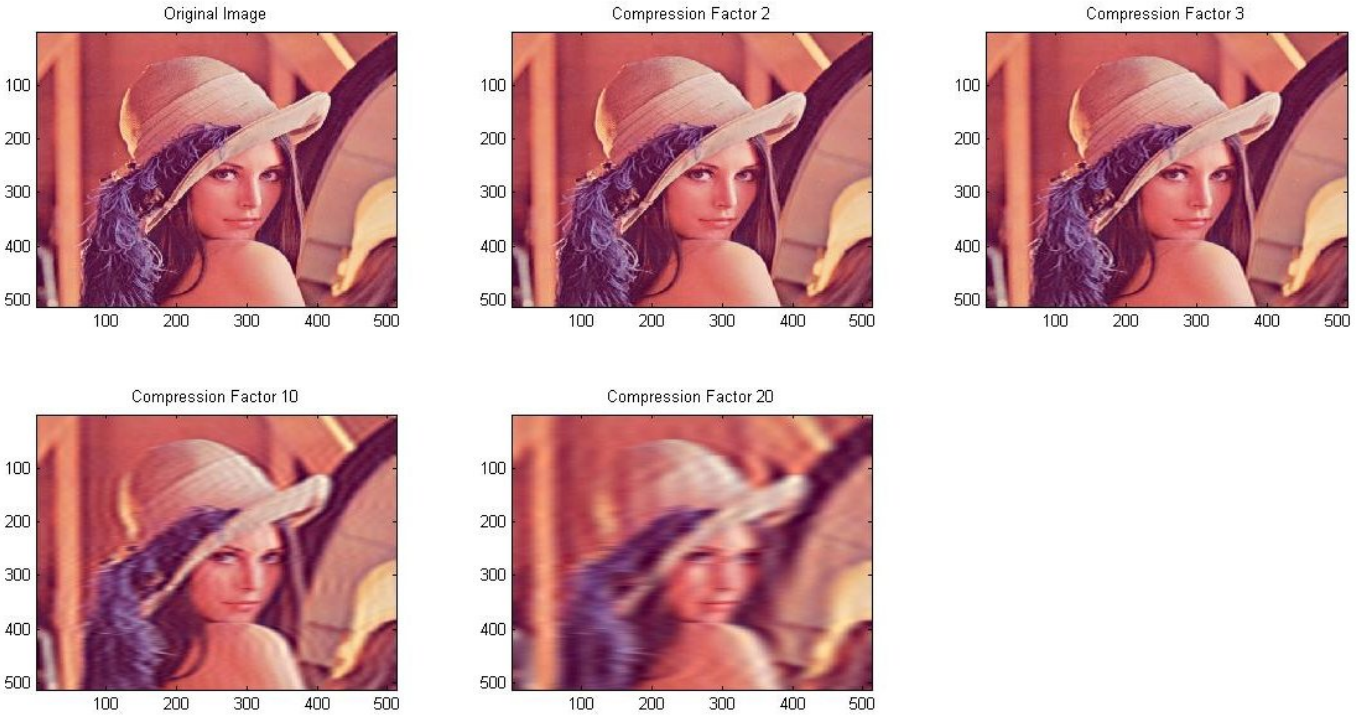

Figure 1: Compression