Description of BlastBrush

Müzeyyennur YILGIN
Department of Engineering,
Computer Engineering
Gazi University
Ankara, Turkey

Ufuk APAYDIN
Department of Engineering,
Computer Engineering
Gazi University
Ankara, Turkey

Kaan YOLA
Department of Engineering,
Computer Engineering
Gazi University
Ankara, Turkey

ABSTRACT

In this paper, we will explain the application we prepared for the final project.

Keywords

Image Editing Tool;PyQt5; Cropping; Effects; Rotation; Flipping

1. INTRODUCTION

When we want to edit a photo part, we use professional tools. These professional tools meet most of our requests with ease. All we have to do is download the application and select the feature we want to apply to the photo. But in a photo editing, things are different in the background. An image is represented by numbers. It is the computer work to change the electronic (digital) image data recorded in the image process in an electronic environment. Image processing is mainly used to modify, alienate or improve existing pictures and graphics that have been saved [1]. In this article, we'll show you what's behind the basic photo editing features.

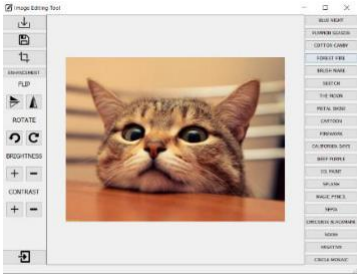
2. TOOLS & TECHNOLOGIES

We designed the entire Graphical User Interface using PyQt5. We also got help from the qt designer application for the design. We made use of cv2, numpy, scipy libraries in the application. The GUI is generally divided into 3 parts:

1. Simple operations and image enhancement: There are buttons where operations such as rotation, cutting and image enhancement can be applied. In addition, there are image upload, image saving and application exit buttons.

2. Image area: The uploaded image and the changes applied to the image appear here.

3. Effects: 20 different effects are in this section.



Simple operations consist of the following buttons:

1. Upload Image

2. Save image

3.Image cropping

4. Image enhancement

5.Image mirroring horizontally and vertically

6. Rotate image left and right

7. Increase and decrease image brightness

8.Increase and decrease image contrast

9. Exiting the application

3. THE APPROACH

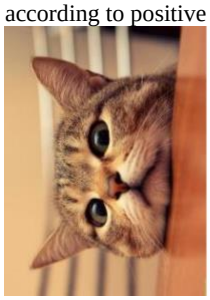
All transactions in our project are explained in detail as follows:

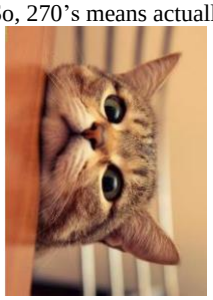
3.1 Basic Operations

The description of the basic operations that we applied in our project is as follows for each of them separately:

3.1.1 Rotation

Our projects one of the main aim is rotate the image. Rotation image process is quite simple in python. First of all, we created a instance QTransform class. Then for right rotation we used transform.rotate(90) function, for left rotation we used transform.rotate(270) function. Because this function transforms according to positive axis. So, 270's means actually -90.





3.1.2 Cropping

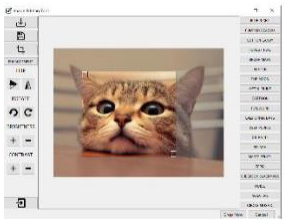
For the clipping process, we first needed to create a cropping box. So, we created the clipping box first. We created a 20x20 space for the user to increase or decrease box by holding it from the

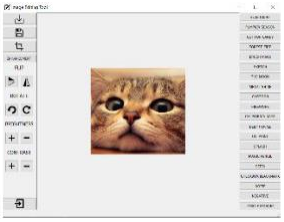
corners. These areas are at the top left and bottom right. We used event to detect mouse movements in this area.

```
if self.clk_area == 1:  # Top left corner is clicked
    new_p1 = self.topleft + moved
    self.p1 = QPoint(max(0, new_p1.x()), max(0, new_p1.y()))

elif self.clk_area == 2:  # Bottom right corner is clicked
    new_p2 = self.btmright + moved
    self.p2 = QPoint(min(self.last_pt.x(), new_p2.x()), min(self.last_pt.y(), new_p2.y()))

elif self.clk_area == 3:  # clicked inside cropbox but none of the corner selected.
    min_dx, max_dx = -self.topleft.x(), self.last_pt.x() - self.btmright.x()
    min_dy, max_dy = -self.topleft.y(), self.last_pt.y() - self.btmright.y()
    dx = max(moved.x(), min_dx) if (moved.x() < 0) else min(moved.x(), max_dx)
    dy = max(moved.y(), min_dy) if (moved.y() < 0) else min(moved.y(), max_dy)
    self.p1 = self.topleft + QPoint(dx, dy)
    self.p2 = self.btmright + QPoint(dx, dy)
```

It can make the crop box larger or smaller, if it holds the upper left corner or lower right corner. If he holds it in the middle of the crushing box, it can move.

The user can give up after pressing the break button. So, we added a bar that is visible after pressing the crop button for this. We have added 'crop now' button to confirm clipping and 'cancel' button to cancel clipping in this bar.





3.1.3 *Flipping*

Our project another main function is flip the image. There are two main axis flip we used. Horizontal flip and vertical flip.

```
def horizontalFlip(self):
    image = self.imageLabel.convert2arrayTemp()
    h, w, _ = image.shape
    temp = np.zeros((h, w, 3), np.uint8)
    for i in range(0, w):
        temp[:, i, :] = image[:, w - i - 1, :]
    image2 = temp
    qimage = QImage(image2, w, h, 3 * w, QImage.Format_BGR888)
    q = QPixmap(qimage)
    self.imageLabel.set_imagePixmap(q)
```

```
def verticalFlip(self):
    image= self.imageLabel.convert2arrayTemp()
    h, w, _= image.shape
    temp = np.zeros((h, w, 3), np.uint8)
    for j in range(0, h):
        temp[j, :, :] = image[h - j - 1, :, :]
    image2 = temp
    qimage=QImage(image2, w,h,3*w, QImage.Format_BGR888)
    q=QPixmap(qimage)
    self.imageLabel.set_imagePixmap(q)
```

3.1.4 *Brightness*

One of the essentials of photo editing applications is brightness adjustment. An image editing program without brightness adjustment is unthinkable. In this section, we will talk about how to adjust the brightness in the picture.

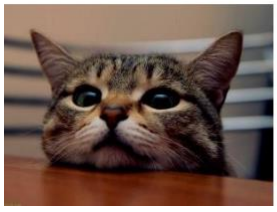
First of all, there are two buttons for brightness. There is plus button to enhance brightness and there is minus button to reduce brightness.

In Python, the brightness process for the picture is quite easy. We can perform this function with one function.

Thanks to cv2.addWeighted function, we can easily perform both brightness enhancement and reduction operations.

To reduce:

```
image = cv2.addWeighted(img, 1, np.zeros(img.shape, img.dtype), 0, -10)
```



To enhance:

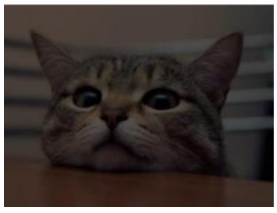
```
image = cv2.addWeighted(img, 1, np.zeros(img.shape, img.dtype), 0,10)
```

3.1.5 *Contrast*

Contrasting the image almost same as brightness. Just it will be some value changes.

To reduce:

```
image = cv2.addWeighted(img, 0.7, np.zeros(img.shape, img.dtype), 0, 0)
```



To enhance:

```
image = cv2.addWeighted(img, 1.25, np.zeros(img.shape, img.dtype), 0, 0)
```

3.2 Image Enhancement

Image editing programs often offer basic automatic image enhancement features that correct color tone and brightness imbalances, as well as other image editing features such as red-eye removal, sharpness adjustments, zooming features, and auto cropping. These are called automatic because they usually happen without user interaction or are offered a single click of a button[2].

Sometimes there is degradation in the images. The causes of distortions are electrical interference, poor lighting, gauss and peppery noises. There are multiple methods to remove these distortions. Image restoration is an objective process, but image enhancement is a subjective process. Image enhancement is adjusting such as contrast, brightness, edge enhancement on the image. In this chapter, we used histogram equalization to automatically improve the image [3].

For image enhancement, we first brought the existing picture. We first separated the color streams. Then we made the histogram equalization of the image using the canned function. Then we converted the image to QImage format to show it on the label.

3.3 Effects

The description of the effects that we applied in our project is as follows for each of them separately:

3.3.1 Blue Night

We first took the original picture to apply this effect. Then we created two tables for use in red and blue images. We used the UnivariateSpline function to create these tables. UnivariateSpline is a smooth univariate spline to fit a specific set of data points [4]. After that we divided the image into colors. We applied the tables created for red and blue color with the LUT function of the cv2 library. LUT means LookUp Table. It means looking at the table or chart. It is a set of matrix code indexing system, consisting of logarithm-style columns and rows [5]. Finally, we combined the modified red and blue image and the green image with the merge function. Since there are blue tones in the effect, we wanted to name it blue night.

3.3.2 Pumpkin Season

We also used the method that we used in the Blue Night effect in this effect. The only difference was the tables that we applied for blue and red images. When we applied the blue night effect in the opposite way, it became more orange and red. That's why we wanted to name this effect pumpkin season.

3.3.3 Cotton Candy

In our third effect, we applied the same method that we applied to the first two effects. However, we created a table only for the red image and we only manipulated the red color. In this way, we obtained a redder and pinker image. That's why we named it cotton candy.

3.3.4 Forest Fire

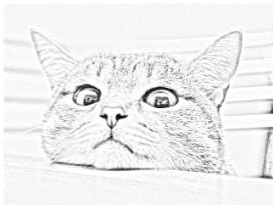
We applied this effect with the same logic. We created different tables for red and blue colors. Green was more prominent in this effect. That's why we mean forest fire.

3.3.5 Brush Mark

Our aim in this effect is to turn our painting into a light graphic image. We used the bilateralFilter function to create this much-loved effect, and we got the result we wanted by adjusting the values of this function 9 300 300.

3.3.6 Sketch

Our aim in this effect is to turn our painting into a beautiful and eye-pleasing black pencil work. We used the GussianBlur function to create this effect, which is often used in various image editing applications. This effect is quite popular.

3.3.7 The Moon

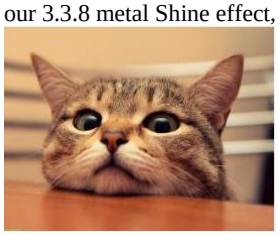
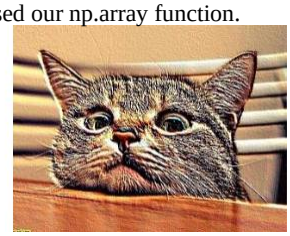
Our aim in this effect is that our picture has an image that feels constantly alive and constantly curious, like the moon, rather than just a black and white image. That's why we called Our effect the moon. Curious as the moon, this filter will be an inspiration, especially for those who want to show photos that are curious. When creating this effect, we colored our image with gray with the cvtColor function.


3.3.8 Metal Shine

Our aim in this effect is for our painting to shine with a metallic sheen and reveal the deepest secrets in it. It's as if photography is asking us what our darkest desires are. We performed this effect with the np.array function.

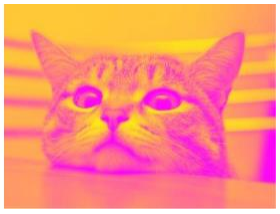

3.3.9 Cartoon

In designing this effect, we were inspired by the cartoons we watched as children. Thanks to our effect, we ensure that very popular cartoons are reflected in today's photos. Again, as with our 3.3.8 metal Shine effect, we used our np.array function.

3.3.10 Firework

In this effect, we aimed to reflect the explosion of fireworks and the uniform image in the sky. In doing so, we achieved my goal by giving the cv2.colormap_spring parameter to the applyColorMap function.

3.3.11 California Days

We tried to reflect the warm beaches of California in this effect. We used two paintings to do that. We created this unique effect by running the LUT function with RGB colors.


3.3.12 Deep Purple

We wanted to create an effect where we focused on the use of purple color, which will also be seen in the photo below. For this, we used the same technique that we used in our 3.3.11 California Days effect, but with different values.


3.3.13 Oil Paint

In this effect, we wanted to give our painting an oil painting appearance. In doing so, we received help from our cv2_morphologyEx and cv2_normalize functions. We've created a successful effect.


3.3.14 Splash

We aimed to provide a splash effect in this effect. First, we have defined the lower December and upper ranges. We have obtained inverting mask using these lower and upper ranges in the bitwise_not function. This inverting mask we obtained for I in range 3 by inserting the gray points into the colored sections have been added.


3.3.15 Magic Pencil

Our aim in this effect is to display visual in the image edges as drawn with different colors. For this effect we apply 3x3 matrix filter shown below. With this filter edge of the visuals on the image starts glow. After the filter we use applied Canny Edge Detection function available from OpenCV to remove excess lines. We dilate this lines with 2x2 matrix consisting of ones to get thicker lines.

$$\begin{matrix} 10 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -10 \end{matrix}$$


3.3.16 Sepia

Our aim in this effect is to make sepia filter that has been popular in other image editing programs. For this effect we use specific matrix that has been use for sepia filter. With this filter we transform matrix our image.


3.3.17 Checkbox Blackmark

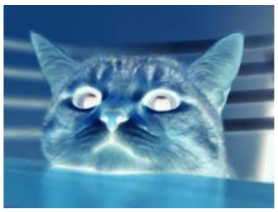
In this effect we try to merge cartoon effect with checkbox tile texture. We use grayscale image for adaptive threshold with mean adaptive threshold method. With this method we get a black and white cartoon like image of the visual. For the second part of this effect we create a new image with black and white checkbox pattern. This pattern tiling count can be arranged in the code. After this the pattern image and cartoon image are multiply to get checkbox cartoon effect and this image added to the original image.


3.3.18 Noise

Our aim in this effect is to display the image with noise like in old the televisions. For this effect we convert image to grayscale and change every pixels intensity. To change intensity, every pixel have %80 chance to get new intensity. If pixel get new intensity, the new intensity will be lower or higher.

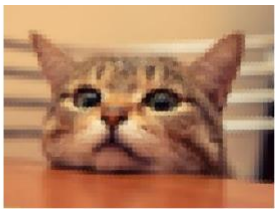

3.3.19 Negative

Our aim in this effect is to convert the image into negative image. To converting image, we use “bitwise_not” function from OpenCV. This function applies bitwise not operator to 8-bit BGR colors in each pixel.

3.3.20 Circle Mosaic

Our aim in this effect is to display image as a mosaic made with circles. To smooth image we use mean filter made from 10x10 matrix consisting of ones. After smoothing image, an ellipse drawn according to the color of each 10 pixels using two nested for loops. The ellipses are arranged to have a radius of 5 pixels so they do not overlap each other.

4. CONCLUSION

In this project, the biggest challenge was converting the picture to cv2 format and then displaying it as a picture again. QPixmap only takes QImage format as variable. The QPixmap which we used to view images caused various problems, such as converting the image from OpenCV format to QImage format. We solved this problem by creating a new function. Another difficulty was that gray images could not be displayed in QImage format. We did research on this subject and fixed this problem by calling in different formats (such as BGR888, ARGB32) according to the picture. We used many methods to create different styles of effects. This situation was also difficult for us. We have obtained new effects by repeating some methods with different numbers. We would like to thank you for our instructor Duygu Sarıkaya for giving this project and giving us information on many issues.

5. REFERENCES

[1] https://tr.wikipedia.org/wiki/G%C3%B6r%C3%BCnt%C3%BC_i%C5%9Fleme

[2] https://en.wikipedia.org/wiki/Image_editing

[3] Maraşlı, Fatih & Ozturk, Serkan. (2018). Görüntü İyileştirme ve Görüntü Onarma Teknikleri ile Yapılmış Uygulamalar

[4] https://docs.scipy.org/doc/scipy/reference/generated/scipy.interpolate.UnivariateSpline.html

[5] http://www.cezmikardas.com/blog/2016/5/17/lut-nedir#:~:text=LUT%20a%C3%A7%C4%B1l%C4%B1m%C4%B1%20ise%20LookUp%20Table,(MATRIX)%20kod%20indeksleme%20sistemidir.