Photoshop using openCV and Python

Abstract

Summarize in one or two paras - What are you trying to solve, what will be the methodology and what was your result

In our work, we have tried to build a web app with photoshop functionalities. The webpage has some basic and advanced image processing modules which consist of a filter module that has 15 sub functionalities/filters and 7 supplementary modules (Converting Image to Sketch, Image Inpainting, Doc Scanner, Adding Titles to images, Cropping an Image, Edge, and contour detection, Face Detection and Feature Detection). All the image filters and other functionalities were developed using different image processing theories and methods like Intensity Transformation, Convolution, Clustering, Masking, etc.

Separate python scripts were created for each of these modules. Finally, we have integrated these different scripts using streamlit which is an open-source python framework for building web apps.

Objectives

precise points - What the project achieves; don't be abstract or vague; For example, if the project identifies all dogs in an image and their breeds, the objectives could be:

a) Segmentation to identify regions associated with dogs
b) Breed recognition of each identified dog
c) Count the number of identifications
d) Draw an enclosing box around the dogs and label it with the breed

Our web app consists of a total of 8 modules. The basic object of each of the modules are given below:

1. **Filter Modules**
   - Bright - This filter increases the brightness of the given image, it uses the HSV(Hue, Saturation, Value) color space
   - Details enhancement- This achieves the sharpening of an image. However, the results obtained are slightly different from what is obtained using the default functionality of cv2.detailEnhance
   - Invert- Using this functionality the pixel values are inverted which means the subtraction of pixel values by 255
   - Summer- In this filter, we have made use of the gamma function. The pixel values are increased for all the gamma values above 1 and decreased for all the gamma values below 1.
   - Winter- In this filter, vice-versa of the summer filter takes place, that is the pixel values are reduced for all the gamma values above 1 and increased if it is below 1
   - Daylight- It basically increases the light of the image by working on the HLS (Hue, Lightness, and Saturation)color space

- 60TVs- This filter converts the image into black and white, thus giving a feeling of 1960s TV channels
- High Contrast- It is a gray-scale filter where the histogram is stretched to the edges to increase the contrast of the input image
- Sepia- This filter gives a warm brownish tone thus improving the general look and feel of your image
- Splash- In splash filter only certain pixel values are changed to grayscale values and other pixel values are kept as such
- Duo-Tone- A screen of a specific color is added to the duo-tone filter, giving the impression that the image was taken with that color of paper against the lens.
- Cartooning- It creates a cartoon image of the input image by maintaining the edges and reducing the number of colors used
- Emboss- Depending on the light/dark limits on the source image, each pixel of an image is replaced by a highlight or a shadow in this filter. A grey backdrop replaces spots with low contrast.
- Pencil Drawing- It is used to create pencil sketches of the input image
- Comic- It creates images similar to those found in comic books

2. **Converting Image to Sketch:** In this module, the input image is initially converted to a grey image and the image is inverted and gaussian blur is applied. And a bit-wise division is done on the image blurred inverted image which results in the generation of the sketch image

3. **Image Inpainting:** The technique of removing damage to images, such as sounds, strokes, or text, is known as image inpainting. It's very excellent for restoring vintage images with scratched edges or ink marks. This procedure can be used to digitally erase these. Image inpainting works by replacing damaged pixels with pixels that are comparable to those in the surrounding pixels, making them less noticeable and allowing them to blend in with the background.

4. **DocScanner:** For this functionality, adaptive thresholding is used. Adaptive thresholding is the method where the threshold value is calculated for smaller regions. This leads to different threshold values for different regions with respect to the change in lighting. It will scan the images of documents the same as traditional document scanners such as CamScanner, Adobe document Scanner, etc.

5. **Adding Titles to Images:**
   By using this functionality we can add headings/titles to the input images. We have used OpenCV's inbuilt functionality called putText() to achieve this task

6. **Cropping an Image:**
   To achieve this task we have used slicing in a numpy array. The first dimension is the number of rows or the height of the image. The second dimension is the number of columns or the width of the image.

7. **Edge and Contour detection**:
   Contour detection is the technique of combining all continuous points with the same color intensity along a border. In the detection and recognition of objects, contour

detection is a useful tool for form analysis. Edge detection, on the other hand, is the technique of detecting boundaries and edges in photographs. Then there's the matter of how contour detection differs from edge detection, as both are eventually employed to determine an object's shape. We have used cv2.Canny function for edge detection while we have initially used binary thresholding and then findContours methods for finding out the contours in the input image

8. **Face Detection:**
   Initially, the input image is converted into grayscale images. There is a detectMultiscale function and faces which contains a list of coordinates for rectangular regions where faces are found

9. **Feature Detection:**
   For feature detection, we have used a haar cascade file. Using the cascadeClassifier we will load the XML file. We have the detectMultiScale function where we pass the image to find the features in that image. If we find the image then draw a rectangle around the feature detection.

Assumptions

1. We can only work on images with formats jpeg,jpg,png etc
2. Some functionalities requires width and height to be predefined
3. Processing time and computation required would be much larger for images with much higher size. Therefore images of lower size are preferred.

System Architecture

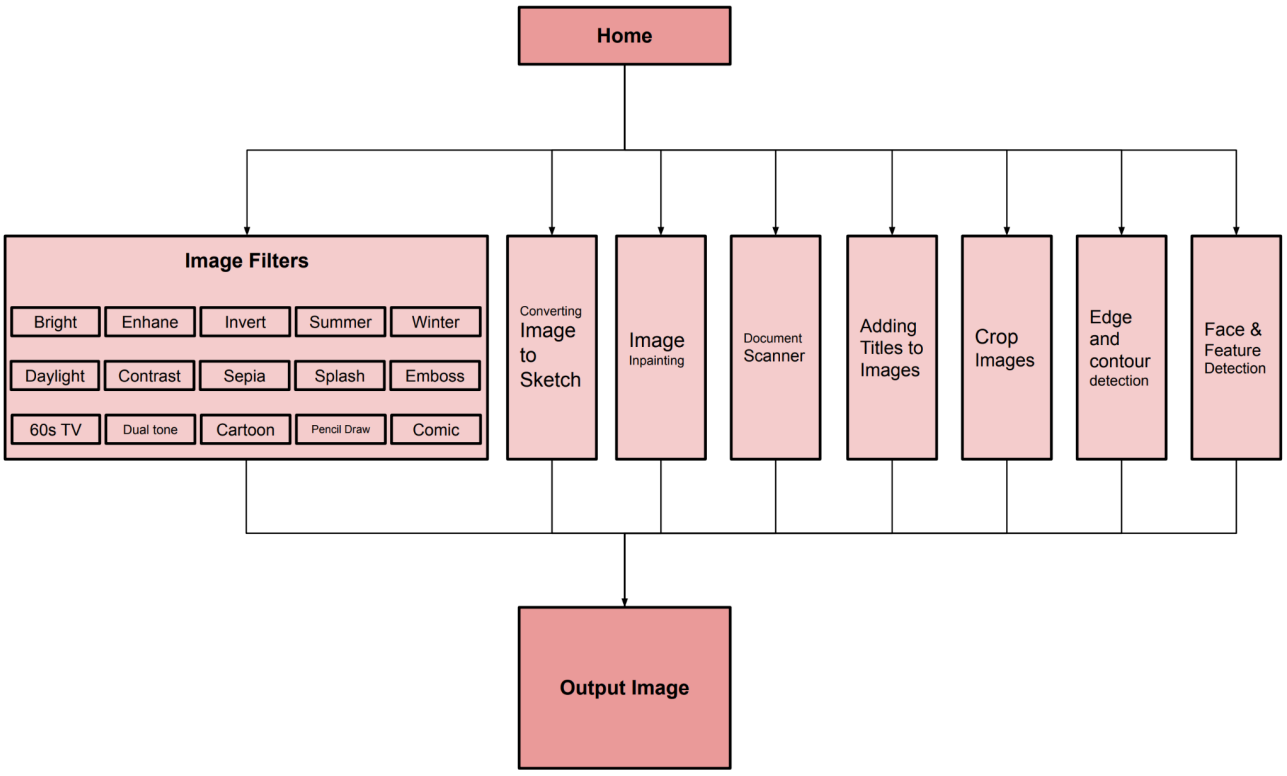


Fig.1. System Architecture

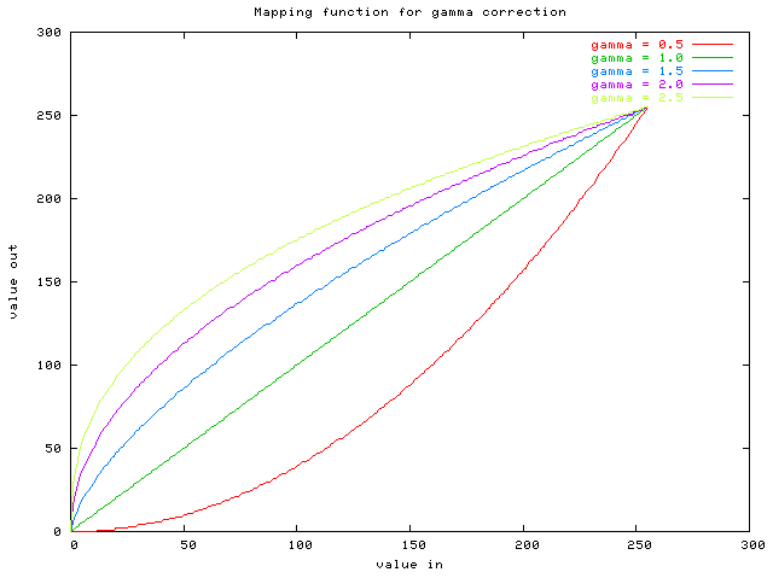
Our web app consists of a total of 8 modules. The basic object of each of the modules are given below:

1. **Filter Modules:**
   - ❖ Bright - This filter increases the brightness of the given image, it uses the HSV(Hue, Saturation, Value) color space. The pixel values of the HSV color image is scaled up for channels 1 and 2. Then we finally convert the image back to BGR color space.

   - ❖ Details enhancement- This achieves the sharpening of an image. However, the results obtained are slightly different from what is obtained using the default functionality of cv2.detailEnhance. The detailEnhance function provided by openCV have parameters InputArray, OutputArray, sigma_s and sigma_r. sigma_s controls how much the image is smoothed - the larger its value, the more smoothed the image gets, but it's also slower to compute. while sigma_r is important if you want to preserve edges while smoothing the image. Small sigma_r results in only very similar colors to be averaged (i.e. smoothed), while colors that differ much will stay intact.

   - ❖ Invert- Using this functionality the pixel values are inverted by applying a bitwise not operation on the image pixels.

   - ❖ Summer- Here we try to increase the impact of the red channel over the other and for this we have used a gamma function. For values of gamma above 1, the values are increased and for below 1, the pixel values are decreased. It is implemented in OpenCV by creating a lookup table and using cv2.LUT() . The values of the Red channel of BGR and Saturation of HSV is increased by taking a value of gamma greater than 1 and the values of the Blue channel is decreased by taking a value less than 1. The graph below shows the gamma values for various channels.



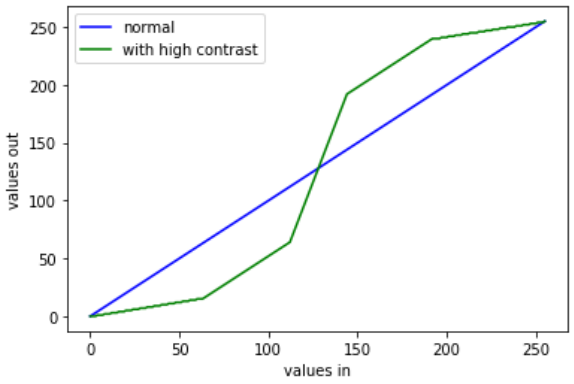
   - ❖ Winter- Here we do the opposite of the summer filter, i.e. we increase the impact of blue channels. For values of gamma above 1, the values are decreased and for below 1, the pixel values are increased. This is also implemented in OpenCV by creating a lookup table and using cv2.LUT().

   - ❖ Daylight- Here we have to increase the light and this can be achieved in the HLS (Hue, Lightness, and Saturation) color space. First we convert the image into HLS color space and then increase the value of the Lightness channel by multiplying it by some value

greater than 1 and finally we convert it back to BGR to get the output.

- 60TVs- Inorder to create this filter we will first convert our image to grayscale and then randomly induce a lot of noise by adding or subtracting a random number and make sure to stay within our limit of 0 to 255.

- High Contrast- It is a gray-scale filter where the histogram is stretched to the edges to increase the contrast of the input image. This can be accomplished with cv2.LUT, and we'll use np.interp to generate a lookup table for piecewise linear interpolation between data points.



- Sepia- This filter gives a warm brownish tone thus improving the general look and feel of your image.What we do is to multiply a special matrix with our image which can be done using cv2.transform and normalize the values above 255 to 255.

- Splash- In a splash filter, only certain pixel values are changed to grayscale values and other pixel values are kept as such. We first create a mask of the image by applying the hsv function to it. The inverse of the mask is created by using the bitwise not operation. Then the regions which are to be the coloured are obtained by applying a bitwise and operation on the original image and the mask. The region to be converted to grayscale is obtained by applying a bitwise and operation on the gray scaled image and the inverted mask. The final output image is obtained by joining the images of both the grayscale and colored components.

- Duo-Tone- A screen of a specific colour is added to the duo-tone filter, giving the impression that the image was taken with that colour of paper against the lens. The exponential function was applied to a single channel, with the values of all other channels set to zero. The change in values in an exponential function is smaller at first and more later, similar to exponents. We have used cv2.LUT which takes an image and a lookup table as the parameters

- Cartooning- It creates a cartoon image of the input image by maintaining the edges and reducing the number of colors used.In essence, we'll employ a succession of filters and picture conversions.To achieve a cartoon effect, we first downscale the image and then apply a bilateral filter. The image is then upscaled once more. The next step is to create a blurred version of the original photograph. We don't want the colours to get in the way of this now. The only thing we want is for the lines to blur. We do this by converting the image to grayscale and then using the media blur filter. The next step is to find the image's edges and combine them with the previously edited images to create a sketch pen look. We're going to use adaptive threshold for this one.

- ❖ Emboss- Depending on the light/dark limits on the source image, each pixel of an image is replaced by a highlight or a shadow in this filter. A grey backdrop replaces spots with low contrast.We have to create a kernel and apply convolution using cv2.conv2D and add a value of 128 to all pixels.

- ❖ Pencil Drawing- It is used to create pencil sketches of the input image.It uses an opencv filter called pencilSketch.This filter produces an output that looks like a pencil sketch. There are two outputs, one is the result of applying the filter to the color input image, and the other is the result of applying it to the grayscale version of the input image

- ❖ Comic- It creates images similar to those found in comic books

2. **Converting Image to Sketch:**
   In this module the input image is initially converted to grey image and the image is inverted and gaussian blur is applied.And a bit-wise division is done on the image blurred inverted image which results in the generation of the sketch image. A histogram equalization is also done to improve the results obtained.

3. **Image Inpainting:**
   The technique of removing damage to images, such as sounds, strokes, or text, is known as image inpainting. It's very excellent for restoring vintage images with scratched edges or ink marks. The original image along with the mask file of the image is inputted to the program. The mask file is used to specify the impainting is to be done. The impainting is done using the cv2.inpaint method. We also specify the algorithm which is to be used for impainting to be the “Fast Marching Method " which is passed as an argument INPAINT_TELEA.

4. **Doc Scanner:**
   For this functionality adaptive thresholding is used. Adaptive thresholding is the method where the threshold value is calculated for smaller regions. This leads to different threshold values for different regions with respect to the change in lighting.

5. **Adding Titles to Images:**
   By using this functionality we can add headings/titles to the input images. We have used opencv’s inbuilt functionality called putText() to achieve this task. This functionality accepts parameters like
   - Image, which is the image on which text will be printed
   - Text, which is the text string that will be drawn.
   - Org is the coordinates of the text string in the image's bottom-left corner.
   - The font type is indicated by the font.
   - FontScale: This is the factor multiplied by the font's base size.
   - Color: This is the colour of the text string that will be drawn.
   - Thickness: This is the line's thickness in pixels.

6. **Cropping an Image:**
   To achieve this task we have used slicing in a numpy array. The first dimension is the number of rows or the height of the image. The second dimension is the number of columns or the width of the image. Inorder to represent this script in streamlit we have used a streamlit package called streamlit_cropper which uses the same numpy slicing

logic.

7. **Edge and Contour detection**:
   Contour detection is the technique of combining all continuous points with the same colour intensity along a border. In the detection and recognition of objects, contour detection is a useful tool for form analysis. Edge detection, on the other hand, is the technique of detecting boundaries and edges in photographs. Then there's the matter of how contour detection differs from edge detection, as both are eventually employed to determine an object's shape. We have used cv2.Canny function for edge detection while we have initially used binary thresholding and then findContours methods for finding out the contours in input image

8. **Face Detection:**
   Initially the input image is converted into grayscale images.There is a detectMultiscale function and faces which contains a list of coordinates for rectangular regions where faces are found.A haar cascade file is used for face detection. We'll load the XML file using the cascadeClassifier. We may use the detectMultiScale function to discover the faces in an image by passing it the image. Draw a rectangle around the face if we discover the image.

9. **Feature Detection:**
   Here we do the feature detection using a haar cascade file and Fast feature detector. Using the cascadeClassifier we will load the XML file. We have the detectMultiScale function where we pass the image to find the features in that image. If we find the image then draw a rectangle around the feature detection.

Contribution of each member of the team

- **Ashwin V:** 5 Filters, Doc Scanner, Image Inpainting, Integrating the scripts with Streamlit
- **Devagopal A M**: 5 Filters, Add Text to Image and Image Cropping, Integrating the scripts with Streamlit
- **Vishal Menon:** 5 Filters, Face Detection, Feature Detection, Image Cropping, Integrating the scripts with Streamlit

Inputs and Outputs:

The inputs and outputs of our various photoshop modules are shown bellow:

Filter Modules:

- **Bright:**
  Input Types Accepted: jpg, jpeg, png

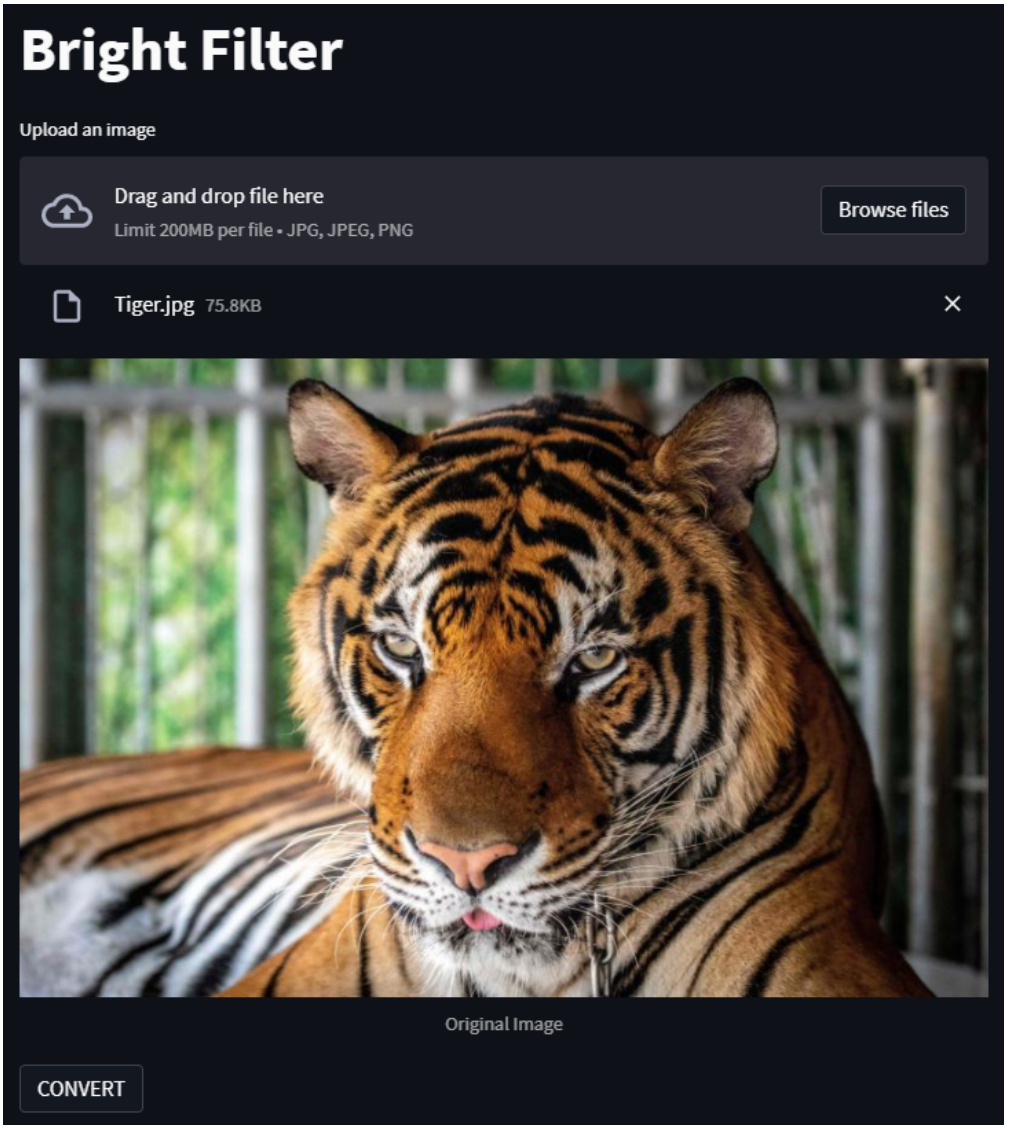


Fig.2.1. Original Image shown in our web app interface

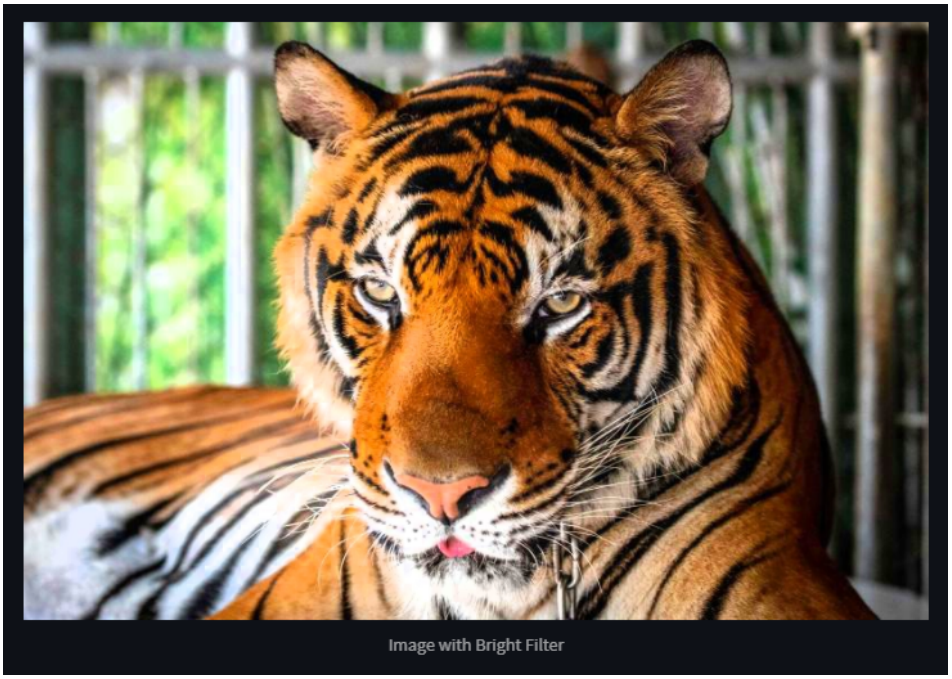


Fig.2.2. Image with Bright Filter

- **<u>Detail Enhancement:</u>**
  Input Types Accepted: jpg, jpeg, png

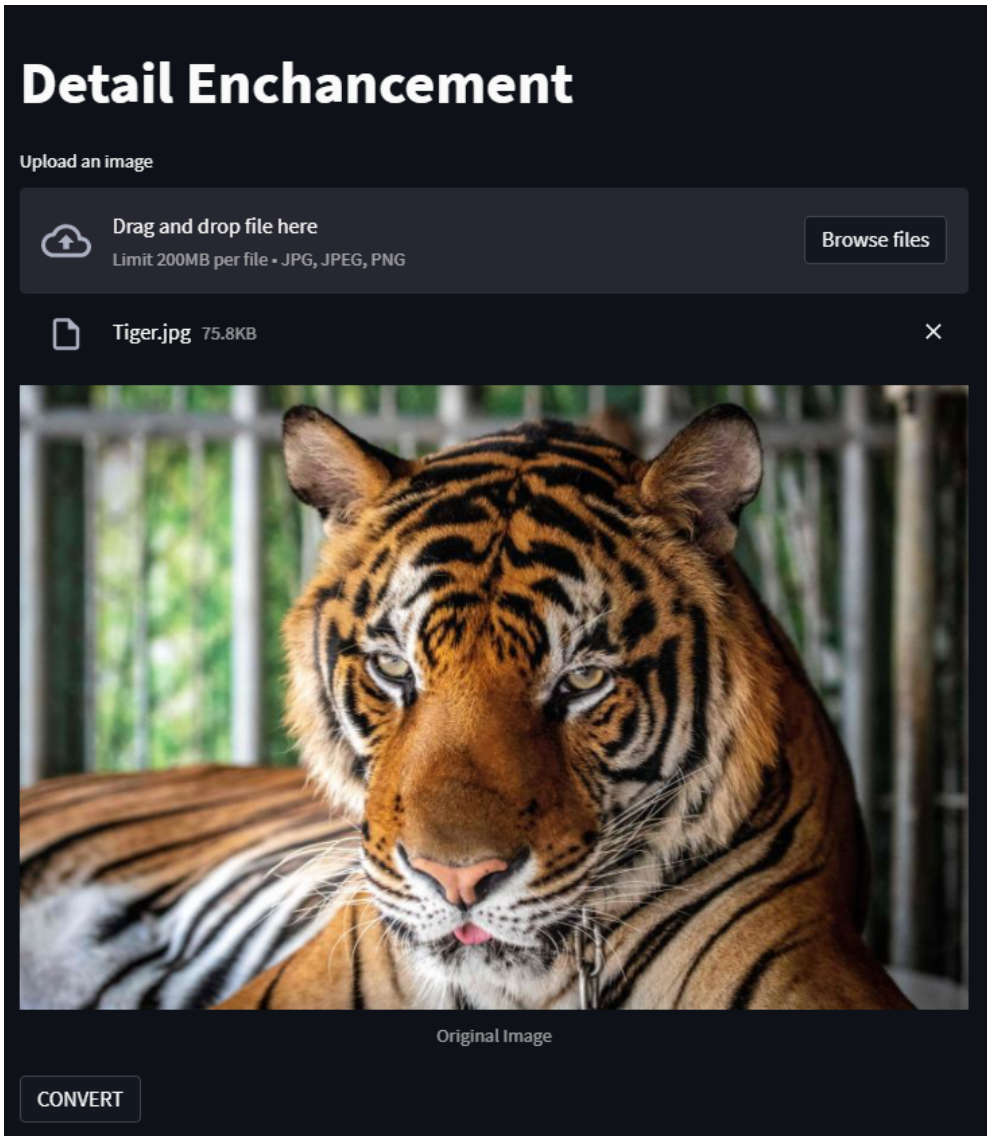


Fig.3.1. Original Image shown in our web app interface

Fig.3.2. Detail enhancement image

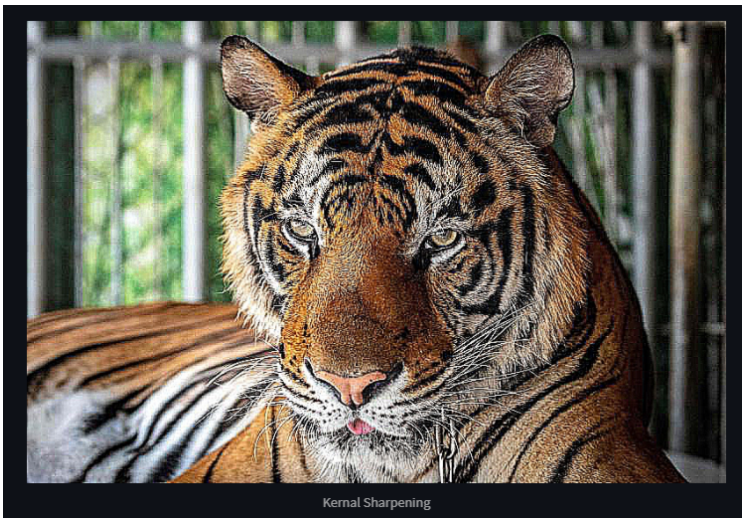


Fig.3.2. Kernal enhancement image

- **Invert:**
  Input Types Accepted: jpg, jpeg, png
  (Webpage interface is same as Bright and Detail enhancement pages)

Fig.4.1. Original

Fig.4.2. Inverted Image

- **Summer:**
  Input Types Accepted: jpg, jpeg, png
  (Webpage interface is same as Bright and Detail enhancement pages)

Fig.5.1. Original

Fig.5.2. Image with Summer Filter

- **Winter:**
  Input Types Accepted: jpg, jpeg, png
  (Webpage interface is same as Bright and Detail enhancement pages)

Fig.6.1. Original

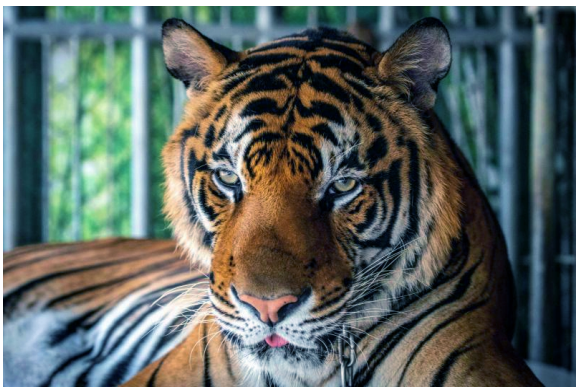


Fig.6.2 Image with Winter Filter

- **Daylight:**
  Input Types Accepted: jpg, jpeg, png
  (Webpage interface is same as Bright and Detail enhancement pages)

Fig.7.1. Original

Fig.7.2. Image with Daylight Filter

- **<u>High Contrast:</u>**
  Input Types Accepted: jpg, jpeg, png
  (Webpage interface is same as Bright and Detail enhancement pages)

Fig.8.1. Original

Fig.8.2. Image with High Contrast Filter

- **<u>Sepia:</u>**
  Input Types Accepted: jpg, jpeg, png
  (Webpage interface is same as Bright and Detail enhancement pages)

Fig.9.1. Original

Fig.9.2. Image with Sepia Filter

- **<u>Splash:</u>**
  Input Types Accepted: jpg, jpeg, png
  (Webpage interface is same as Bright and Detail enhancement pages)
  <u>Note:</u> The splash filter only works successfully for images with objects having high contrast colors (Eg: Yellow and Blue). The image given below is a good example over which the splash filter works successfully.

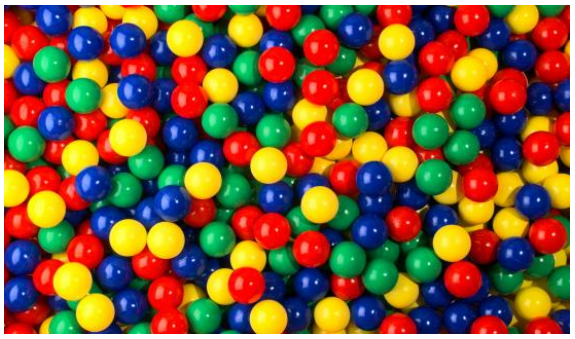


Fig.10.1. Original

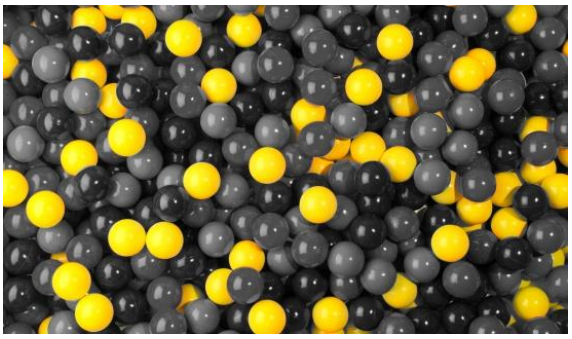


Fig.10.2. Image with Splash Filter

- **Emboss:**
  Input Types Accepted: jpg, jpeg, png
  (Webpage interface is same as Bright and Detail enhancement pages)

Fig.11.1. Original

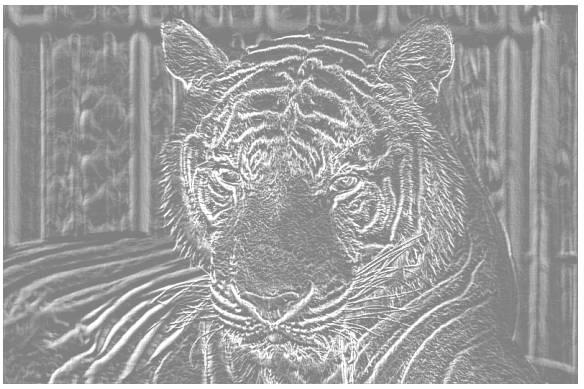


Fig.11.2. Image with Emboss Filter

- **60s TV:**
  Input Types Accepted: jpg, jpeg, png
  (Webpage interface is same as Bright and Detail enhancement pages)
  Note: This filter also consists of 2 other input parameters the noise and threshold values. Our Webpage provides a slider widget to set these parameter values seamlessly. The image example given below shows the interface and outputs.

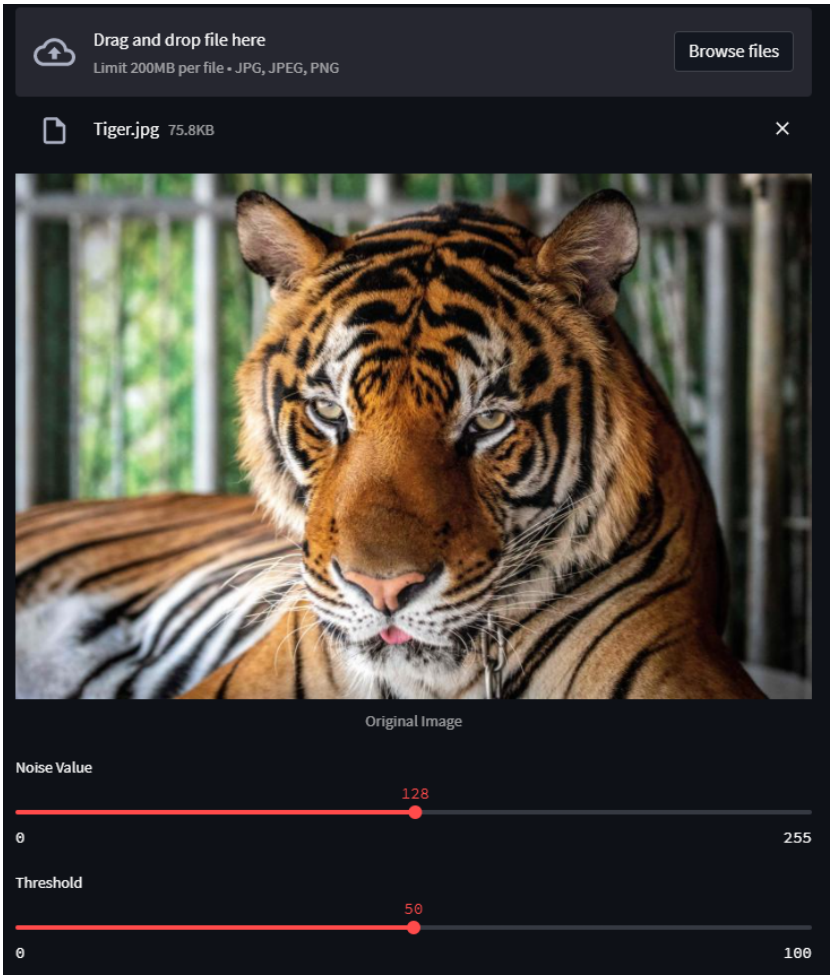


Fig.12.1. Original Image shown in our web app interface

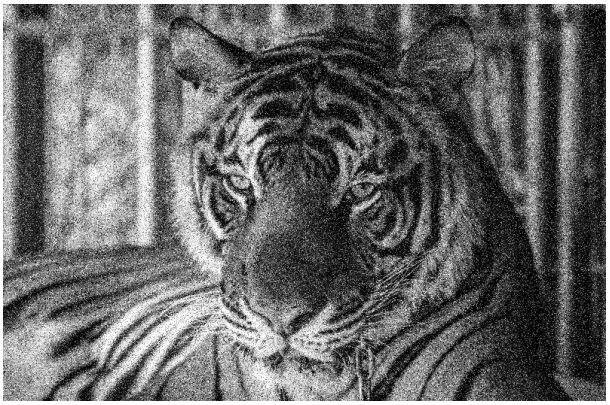


Fig.12.1. Image with 60s TV filter

- **Dual Tone:**
  Input Types Accepted: jpg, jpeg, png
  (Webpage interface is same as Bright and Detail enhancement pages)

Fig.13.1. Original

Fig.13.2. Image with Dual Tone in Red Channel

Fig.13.3. Image with Dual Tone in Green Channel

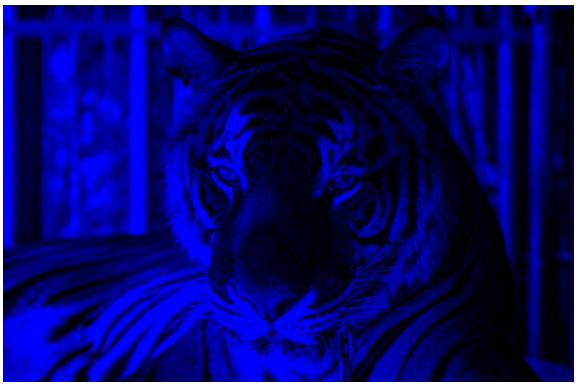


Fig.13.4. Image with Dual Tone in Red Channel

- **Drawing Filter:**
  Input Types Accepted: jpg, jpeg, png
  (Webpage interface is same as Bright and Detail enhancement pages)

Fig.14.1. Original

Fig.14.2. Image with Drawing Filter

- **Pencil Drawing:**
  Input Types Accepted: jpg, jpeg, png
  (Webpage interface is same as Bright and Detail enhancement pages)

Fig.15.1. Original

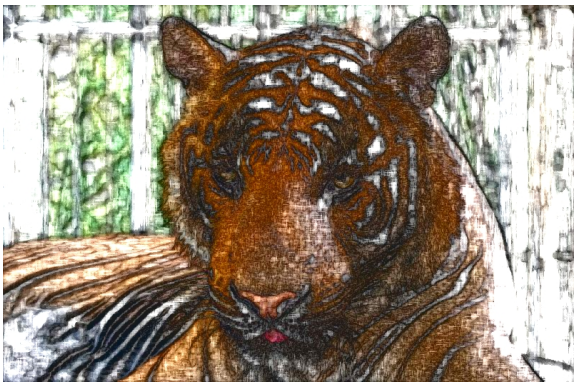


Fig.15.2. Image with Pencil Drawing Filter

- **Comic (Using K-Means):**
  Input Types Accepted: jpg, jpeg, png
  (Webpage interface is same as Bright and Detail enhancement pages)

Fig.16.1. Original

Fig.16.2. Comic Effect when k = 5

Fig.16.3. Comic Effect when k = 8

Fig.16.4. Comic Effect when k = 15

Image to Sketch Module:
Input Types Accepted: jpg, jpeg, png
(Webpage interface is same as Bright and Detail enhancement pages)

Fig.17.1. Original

Fig.17.2. Image converted to Sketch

Image inpainting Module:

Input Types Accepted: jpg, jpeg, png

(Webpage interface is same as Bright and Detail enhancement pages)

Note: This module takes 2 images as its inputs: the first benign our original image and the second is the mask image of the section to be removed or inpainted over. The images given below shows the type of input images and the output generated.

Fig.18.1. Original Image

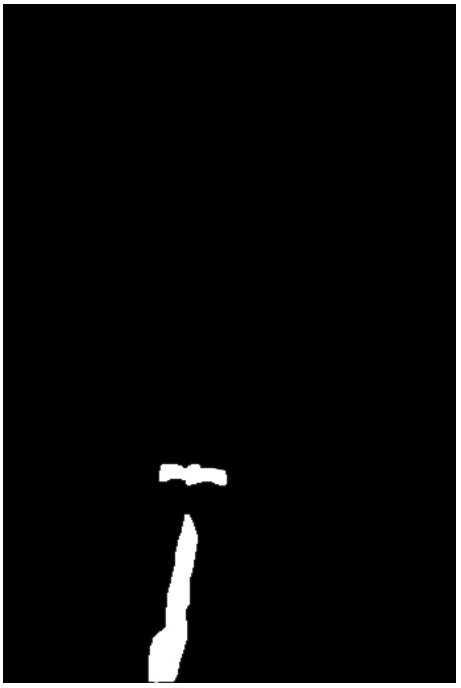


Fig.18.2. Mask Image

Fig.18.3. Inpainted Image

Doc Scanner Module:

Input Types Accepted: jpg, jpeg, png

(Webpage interface is same as Bright and Detail enhancement pages)

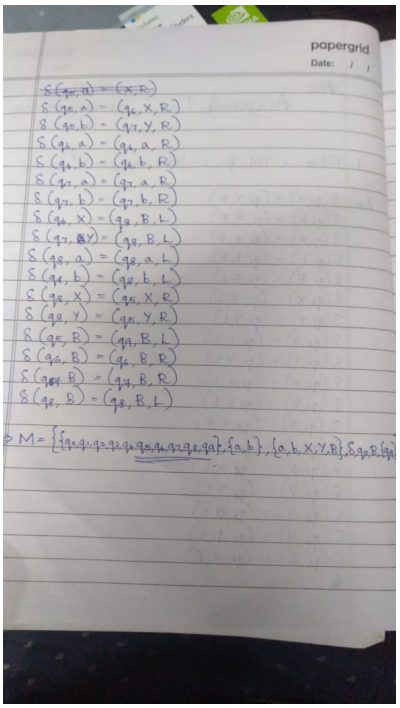


Fig.19.1. Original Doc Image

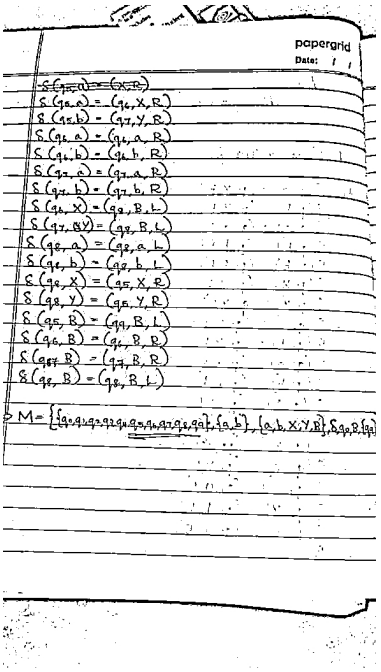


Fig.19.2. Enhanced Doc Image

Add Title to Image Module:

Input Types Accepted: jpg, jpeg, png

(Webpage interface is same as Bright and Detail enhancement pages)

Fig.20.1. Original Image

Fig.20.2. Image after adding Title

Edge and Contour Detection Module:

Input Types Accepted: jpg, jpeg, png

Fig.21.1. Original Image

Fig.21.2. Canny Edge detected Image

Fig.21.3. Contour detected Image

Face Detection Module:

Input Types Accepted: jpg, jpeg, png

(Webpage interface is same as Bright and Detail enhancement pages)

Fig.22.1. Original Image

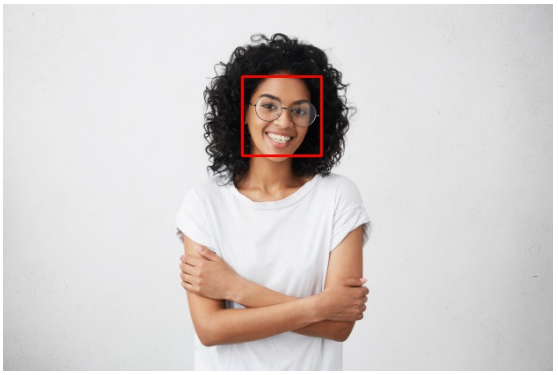


Fig.22.2. Face detected Image

Feature Detection using Fast Featur Detector Module:

Input Types Accepted: jpg, jpeg, png

(Webpage interface is same as Bright and Detail enhancement pages)

Fig.23.1. Original Image

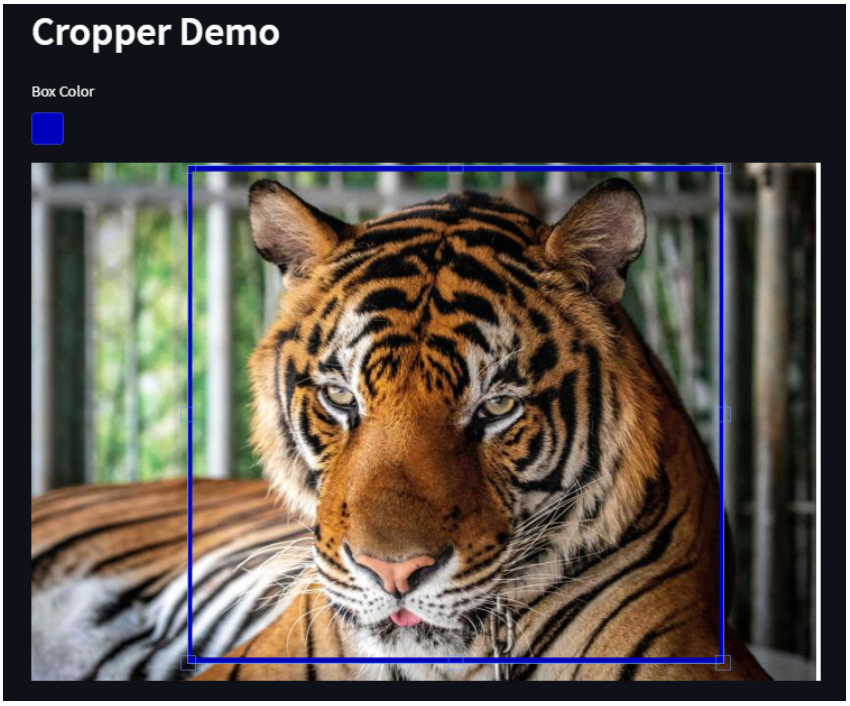
Fig.23.2. Key Points Detected

Fig.23.3 Features Detected using Fast Feature detector

Cropping Image Module:

Input Types Accepted: jpg, jpeg, png

(Webpage interface is same as Bright and Detail enhancement pages)


Cropper Demo
Box Color