JSS SCIENCE AND TECHNOLOGY UNIVERSITY
MYSORE-570006

Live Sketching Application

Mini project report submitted in partial fulfillment of curriculum prescribed for the Digital Image Processing *(*CS552*)* course for the award of the degree of

BACHELOR OF ENGINEERING IN
COMPUTER SCIENCE AND ENGINEERING

By

USN	NAME	ROLL.NO
01JST18CS047	Kartik Nagaraj Nayak	17
01JST18CS099	Rajath R Rao	30
01JST18CS106	Rishitha Ramesh	32
01JST18CS133	Shriya Mittapalli	40

Submitted to ,

Prof. Shruthi N M

Assistant Professor

Dept of CS&E, SJCE Mysore

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

ABSTRACT

We live in an increasingly digital world where information is always right at our fingertips. We expect it to be fast, high quality, and convenient. Perhaps more than anything, our digital world means more and more information is being presented in a visual format. Live Sketching software allows users to create a drawing with digital graphics. Instead of traditional art supplies such as paper and pen, artists create drawings on a computer or tablet using digital tools such as brush sets, color palettes, and more. These drawings can range from 2D drafts that visually communicate how something functions or is constructed to digital artwork. In this project report, we have built a python-opencv program to make live sketch from the webcam live feed.

CONTENTS

1 Introduction

Digital image processing is the use of a digital computer to process digital images through an algorithm. As a subcategory or field of digital signal processing, digital image processing has many advantages over analog image processing. It allows a much wider range of algorithms to be applied to the input data and can avoid problems such as the build-up of noise and distortion during processing. Since images are defined over two dimensions (perhaps more) digital image processing may be modeled in the form of multidimensional systems.

Given the real time webcam data, this paint-like python application uses OpenCV library to show a live sketch of webcam feed which makes it both awesome and challenging.

2 Applications

1. These products can be used by illustrators or designers in many different industries, such as automotive design and traditional construction.

2. These tools allow you to take sketches one step further with functions such as pushing and pulling surfaces to turn them into 3D designs. Sketching solutions typically integrate with general-purpose CAD, as sketching files may be imported to CAD tools to create 3D models.

3. We can make sketch books, create content, build apps which bring professional paint and drawing tools to artists and novices.

4. Widely used in conceptual design for film, television and video games.

5. Live sketching lends itself to a multimedia format because it can potentially be viewed in many ways, including on TV and the Internet, on computers, and on multiple social media platforms

3 Tools and Technologies used

3.1 Python

Python is the programming language used in the project. Python provides a lot of libraries that are very helpful for image processing, including the ones mentioned below.

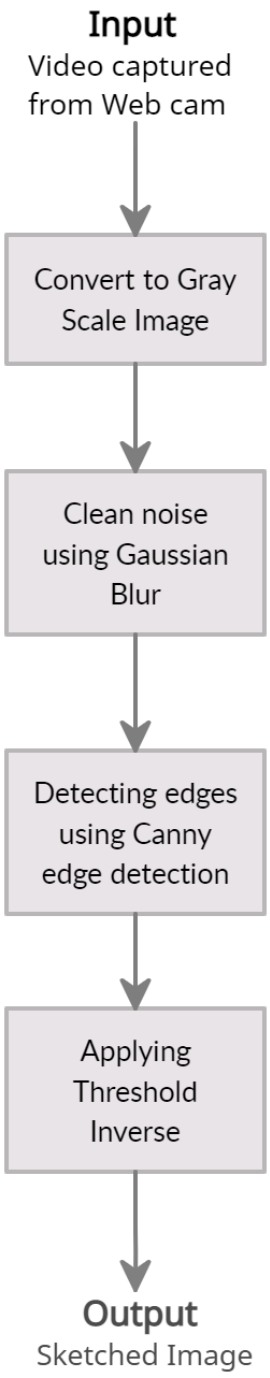
3.2 Open CV

OpenCV is used for developing real-time computer vision applications. It is basically used for image processing, video capture and analysis including features like object detection and face detection. The library consists of more than 2500 optimized algorithms.

3.3 Numpy

NumPy is a python library that is used for working with arrays. It has functions for working in domain of Fourier transform, linear algebra and matrices. NumPy provides an array object that is up to 50 times faster than traditional python lists. NumPy is also an efficient multi-dimensional container of generic data. Arbitrary data types can be characterized using Numpy which permits NumPy to consistently and rapidly integrate with databases.

4 Architectural Design



5 Implementation

1. Since it's a live sketch, we need to use the webcam and extract image frames from the video

2. **Covert the image to grayscale image**
   We are interested in detecting white or yellow lines on images, which show a particularly high contrast when the image is in grayscale. The conversion from RGB to a different space helps in reducing noise from the original three color channels. This is a necessary pre-processing steps before we can run more powerful algorithms to isolate lines.

3. **Next blur the image using Gaussian Blur to clean the noise.**
   In this project we are using Gaussian blur - it saves the edges to an extent and blurs the rest of the image. Here (5,5) is the kernel size ie, the matrix of pixels over which blurring is performed. More is the number in the brackets more will be the blurring effect.

4. **Detect edges in the image using Canny edge**.

We use canny edge method as it is Optimal due to low error rate, well defined edges and accurate detection.

The OpenCV implementation requires passing in two parameters in addition to our blurred image, a low and high threshold which determine whether to include a given edge or not. A threshold captures the intensity of change of a given point (you can think of it as a gradient). Any point beyond the high threshold will be included in our resulting image, while points between the threshold values will only be included if they are next to edges beyond our high threshold. Edges that are below our low threshold are discarded.

5. **Invert the threshold as a finishing touch**. Here ,Threshold Value if the value of intensity after which the pixel will become white. Below the threshold all pixels will be black.

   But, if the threshold type is Threshold Inverse then the scenario will be opposite.

INPUT

OUTPUT

Code:

Live sketch

Import required libraries

In [1]:
```
import cv2
import numpy as np
#from matplotlib import pyplot as plt
```

Function to generate Sketch

In [2]:
```
def sketch(image):
 #convert image to grayscale
 img_gray=cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)
 #cleaning up the image using Gaussian blur
 img_gray_blur=cv2.GaussianBlur(img_gray,(5,5),0)
 #extract edges
 canny_edges=cv2.Canny(img_gray_blur,10,70)
 #do an invert binarize the image
 ret, mask=cv2.threshold(canny_edges,70,255,cv2.THRESH_BINARY_INV)
 return mask
```

Capture image from webcam

In [3]:
```
cap=cv2.VideoCapture(0)
while True:
 ret,frame=cap.read()
 cv2.imshow('livesketcher',sketch(frame))
 if cv2.waitKey(1) == 13:
  break
```

In [4]:
```
cap.release()
cv2.destroyAllWindows()
```

5 Conclusion

In this project, implementation of Digital Live Sketching of our webcam feed is discussed. It is a simple OpenCV application which uses numpy and cv2 to manipulate images. We have introduced several layers of image processing which combine together to give an output. Similarly, solutions can be tailored in accordance with the specific problems and conditions involved; Various applications can be built with this scheme to overcome the current challenges of live sketching and for further advancement in this domain.

7 References

i. Digital Image Processing 4th edition Rafael C. Gonzalez , Richard E. Woods
ii. https://anaconda.org/conda-forge/opencv
iii. https://pypi.org/
iv. https://docs.anaconda.com/
v. https://stackoverflow.com/
vi. https://jupyter.org/install