

TRAINING REPORT
At
SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

Submitted in partial fulfillment of the requirements for the award
of Bachelor of Engineering Degree in

Computer Science and Engineering

By
Kumar Aditya (38110276)



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING
SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY
JEPPIAAR NAGAR, RAJIV GANDHI SALAI,
CHENNAI – 600119, TAMILNADU

March 2022



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

Accredited with Grade “A” by NAAC

(Established under Section 3 of UGC Act, 1956)

JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI– 600119

www.sathyabamauniversity.ac.in



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **Kumar Aditya (38110276)** who carried out the project entitled “**Virtual paint ecosystem using NumPy and Open CV**” under my supervision from Jan 2022 to March 2022.

Internal Guide

Ms. R. Aishwarya

Head of Department

Dr. S.VIGNESHWARI, M.E., Ph.D.,

Dr. Lakshmanan L, M.E, Ph.D.,

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

I, **Kumar Aditya (38110276)** hereby declare that the Project Report entitled “**Virtual paint ecosystem using NumPy and Open CV**” done by me under the guidance of **Ms. R. Aishwarya** at Sathyabama Institute of Science and Technology is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering.

DATE:

PLACE: CHENNAI

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management** of **SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala M.E., Ph.D., Dean**, School of Computing, **Dr. S. Vigneshwari M.E., Ph.D.,** and **Dr. L. Lakshmanan M.E., Ph.D.,** Heads of the Department of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Ms. R. Aishwarya** for his valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

On-Air painting is one of the many attractive and demanding research field in domain of pattern identification and image rectification. It contributes immensely to a new human-machine interaction application. The traditional art of writing in the age of digital era is being replaced by digital art. For creating, rectifying, and visualizing paint in air, we are using hand gesture recognition with the use of machine learning algorithm by using python programming, which creates natural interaction between man and machine and allows the user to draw in the air in a natural way. Paint software have been in our life since 1985 with the first release of windows 1.0. Artists use paint to create beautiful canvas. With the profound growth in technology, new methods and devices have come up but requirement of another input device like mouse or a pencil are still there. We propose unified CNN-RNN approach for shade detection and segmentation strategies to accomplish this goal. Color detection is a shade handling strategy where we can identify any tone in each scope of HSV shading space which coordinates the advantages of both CNNs (Convolutional Neural Networks) and RNNs (Recurrent Neural Networks).

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
1	INTRODUCTION	9
	1.1 OUTLINE	9
	1.2 WHY USE PYTHON, NUMPY, PANDAS	9
	1.3 WHY USE OPEN CV, TKINTER	10
	1.4 LITERATURE REVIEW	11
	1.5 PROBLEM STATEMENT	13
	1.6 OBJECTIVES	13
2	AIM & SCOPE OF FACE RECOGNITION	14
	2.1 REQUIREMENTS	14
	2.1.1 HARDWARE REQUIREMENTS	14
	2.1.2 SOFTWARE REQUIREMENTS	14
	2.2 ROLE OF VISUAL STUDIO CODE	14
	2.2.1 LANGUAGES USED FOR PROJECT	15
	2.3 INSTALLATION OF PYTHON	17
	2.4 INSTALLATION OF VISUAL STUDIO CODE	19
	2.6 INSTALLING PYTHON MODULES	22
4	METHODS AND MATERIAL USED	38
	4.1 MATERIAL USED	38

	4.2 METHODS	38
	4.2.1 DESIGNING THE PROJECT	38
4	RESULT AND DISCUSSION	43
5	CONCLUSION AND FUTURE WORK	45
	REFERENCES	32
	SCREENSHOTS	34

LIST OF FIGURES

- I. Python website to download Python
- II. How Python works
- III. Installing packages
- IV. Open CV
- V. Training Data
- VI. Result

CHAPTER 1

INTRODUCTION

1.1 OUTLINE

The traditional art of writing in the age of digital era is being replaced by digital art. Digital art refers to the forms of speech and the transmission of art in digital form. Tracking an object is viewed as a significant importance inside the field of Computer Vision. The innovation of fast computers, the accessibility of affordable and excellent video cameras and the requirements of computerized video examination have given rise to importance in tracking methods. Free-to-air collaboration is a hotly debated topic in development of communication and release of consumer-level computer platforms such as Microsoft Kinect and other movement tracking advancements. In spite of recent advances in object detection and tracking, accurate and robust detection and tracking of the fingertip remains a challenging task, primarily due to small dimension of the fingertip. Moreover, the initialization and termination of mid-air finger writing is also challenging due to the absence of any standard delimiting criterion. To solve these problems, we propose a new writing hand pose detection algorithm for initialization of air-writing using the Faster R-CNN framework for accurate hand detection followed by hand segmentation and finally counting the number of raised fingers based on geometrical properties of the hand. Further, we propose a robust fingertip detection and tracking approach using a new signature function called distance-weighted curvature entropy. Finally, a fingertip velocity-based termination criterion is used as a delimiter to mark the completion of the air-writing gesture. Experiments show the superiority of the proposed fingertip detection and tracking algorithm over state-of-the-art approaches giving a mean precision of 73.1% while achieving real-time performance at 18.5 fps, a condition which is of vital importance to air-writing.

1.2 WHY USE PYTHON, NUMPY, PANDAS

Python is open source, interpreted, high level language and provides great approach

for object-oriented programming. It is one of the best languages used by data scientist for various data science projects/application. It provides great functionality to deal with mathematics, statistics and scientific function. It provides great libraries to deals with data science application. It uses the elegant syntax; hence the programs are easier to read. The interactive mode of Python makes its simple to test codes. It allows developer to run the code anywhere, including Windows, Mac OS X, UNIX, and Linux. It is free software in a couple of categories. It does not cost anything to use or download Python or to add it to the application.

Numpy is Python library that provides mathematical function to handle large dimension array. It provides various method/function for Array, Metrics, and linear algebra. NumPy stands for Numerical Python. It provides lots of useful features for operations on n-arrays and matrices in Python. The library provides vectorization of mathematical operations on the NumPy array type, which enhance performance and speeds up the execution. It's very easy to work with large multidimensional arrays and matrices using NumPy. Nearly every scientist working in Python draws on the power of NumPy. NumPy brings the computational power of languages like C and Fortran to Python, a language much easier to learn and use. With this power comes simplicity: a solution in NumPy is often clear and elegant. The core functionality of NumPy is its "ndarray", for *n*-dimensional array, data structure. These arrays are strided views on memory.

Pandas is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language. Pandas is one of the most popular Python library for data manipulation and analysis. Pandas provide useful functions to manipulate large amount of structured data. Pandas provide easiest method to perform analysis. It provides large data structures and manipulating numerical tables and time series data. Pandas is a perfect tool for data wrangling. Pandas is designed for quick and easy data manipulation, aggregation, and visualization. There two data structures in Pandas: Series handle and store data in one-dimensional data whereas DataFrame handle and store Two-dimensional data. It is a multi-paradigm programming language. Object-oriented programming and structured programming are fully supported, and many of its features support

functional programming and aspect-oriented programming (including by meta programming and metaobjects (magic methods)). Many other paradigms are supported via extensions, including design by contract and logic programming.

It is a group by engine allowing split-apply-combine operations on data sets.

Data science is an interdisciplinary field focused on extracting knowledge from data sets, which are typically large (see big data). The field encompasses analysis, preparing data for analysis, and presenting findings to inform high-level decisions in an organization. As such, it incorporates skills from computer science, mathematics, statistics, information visualization, graphic design, complex systems, communication and business. Data science is a "concept to unify statistics, data analysis, machine learning, domain knowledge and their related methods" in order to "understand and analyze actual phenomena" with data. NumPy targets the CPython reference implementation of Python, which is a non-optimizing bytecode interpreter. Mathematical algorithms written for this version of Python often run much slower than compiled equivalents. NumPy addresses the slowness problem partly by providing multidimensional arrays and functions and operators that operate efficiently on arrays; using these requires rewriting some code, mostly inner loops, using NumPy. The core functionality of NumPy is its "ndarray", for n-dimensional array, data structure. These arrays are strided views on memory.

Inserting or appending entries to an array is not as trivially possible as it is with Python's lists. The `np.pad(...)` routine to extend arrays actually creates new arrays of the desired shape and padding values, copies the given array into the new one and returns it. NumPy's `np.concatenate([a1,a2])` operation does not actually link the two arrays but returns a new one, filled with the entries from both given arrays in sequence. Reshaping the dimensionality of an array with `np.reshape(...)` is only possible as long as the number of elements in the array does not change. These circumstances originate from the fact that NumPy's arrays must be views on contiguous memory buffers. A replacement package called Blaze attempts to overcome this limitation. Algorithms that are not expressible as a vectorized operation will typically run slowly because they must be implemented in "pure Python", while vectorization may increase memory complexity of some operations from constant to

linear, because temporary arrays must be created that are as large as the inputs. Runtime compilation of numerical code has been implemented by several groups to avoid these problems; open source solutions that interoperate with NumPy include `scipy.weave`, `numexpr`. Many modern large-scale scientific computing applications have requirements that exceed the capabilities of the NumPy arrays. For example, NumPy arrays are usually loaded into a computer's memory, which might have insufficient capacity for the analysis of large datasets. Further, NumPy operations are executed on a single CPU. However, many linear algebra operations can be accelerated by executing them on clusters of CPUs or of specialized hardware, such as GPUs and TPUs, which many deep learning applications rely on. As a result, several alternative array implementations have arisen in the scientific python ecosystem over the recent years, such as Dask for distributed arrays and TensorFlow or JAX for computations on GPUs. Because of its popularity, these often implement a subset of Numpy's API or mimic it, so that users can change their array implementation with minimal changes to their code required.[6] A recently introduced library named CUPy, accelerated by Nvidia's CUDA framework, has also shown potential for faster computing, being a 'drop-in replacement' of NumPy.

1.3 WHY USE OPEN CV, TKINTER

OpenCV is the huge open-source library for the computer vision, machine learning, and image processing and now it plays a major role in real-time operation which is very important in today's systems. By using it, one can process images and videos to identify objects, faces, or even handwriting of a human. When it integrated with various libraries, such as NumPy, python can process the OpenCV array structure for analysis. To Identify image pattern and its various features we use vector space and perform mathematical operations on these features.

The first OpenCV version was 1.0. OpenCV is released under a BSD license and hence it's free for both academic and commercial use. It has C++, C, Python and Java interfaces and supports Windows, Linux, Mac OS, iOS and Android. When OpenCV was designed the main focus was real-time applications for computational efficiency. All things are written in optimized C/C++ to take advantage of multi-core

processing.

Tkinter provides Python users with a simple way to create GUI elements using the widgets found in the Tk toolkit. Tk widgets can be used to construct buttons, menus, data fields, etc. in a Python application. Once created, these graphical elements can be associated with or interact with features, functionality, methods, data or even other widgets.

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. Being a BSD-licensed product, OpenCV makes it easy for businesses to utilize and modify the code.

The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc. OpenCV has more than 47 thousand people of user community and estimated number of downloads exceeding 18 million. The library is used extensively in companies, research groups and by governmental bodies.

Along with well-established companies like Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda, Toyota that employ the library, there are many startups such as Applied Minds, VideoSurf, and Zeitera, that make extensive use of OpenCV. OpenCV's deployed uses span the range from stitching streetview images together, detecting intrusions in surveillance video in Israel, monitoring mine equipment in China, helping robots navigate and pick up objects at Willow Garage, detection of swimming pool drowning accidents in Europe, running interactive art in Spain and

New York, checking runways for debris in Turkey, inspecting labels on products in factories around the world on to rapid face detection in Japan.

It has C++, Python, Java and MATLAB interfaces and supports Windows, Linux, Android and Mac OS. OpenCV leans mostly towards real-time vision applications and takes advantage of MMX and SSE instructions when available. A full-featured CUDA and OpenCL interfaces are being actively developed right now. There are over 500 algorithms and about 10 times as many functions that compose or support those algorithms. OpenCV is written natively in C++ and has a templated interface that works seamlessly with STL containers.

The tkinter package (“Tk interface”) is the standard Python interface to the Tcl/Tk GUI toolkit. Both Tk and tkinter are available on most Unix platforms, including macOS, as well as on Windows systems.

Running `python -m tkinter` from the command line should open a window demonstrating a simple Tk interface, letting you know that tkinter is properly installed on your system, and also showing what version of Tcl/Tk is installed, so you can read the Tcl/Tk documentation specific to that version.

Tkinter supports a range of Tcl/Tk versions, built either with or without thread support. The official Python binary release bundles Tcl/Tk 8.6 threaded. See the source code for the `_tkinter` module for more information about supported versions.

Tkinter is not a thin wrapper, but adds a fair amount of its own logic to make the experience more pythonic.

For example, a button widget can accept mouse clicks, and can also be programmed to perform some kind of action, such as exiting the application.

1.4 LITERATURE REVIEW

Python was conceived in the late 1980s by Guido van Rossum at Centrum Wiskunde & Informatica (CWI) in the Netherlands as a successor to the ABC language (itself inspired by SETL), capable of exception handling and interfacing with the Amoeba operating system. Its implementation began in December 1989. Van Rossum shouldered sole responsibility for the project, as the lead developer, until 12 July 2018, when he announced his "permanent vacation" from his responsibilities as Python's Benevolent Dictator For Life, a title the Python community bestowed upon him to reflect his long-term commitment as the project's chief decision-maker. He now shares his leadership as a member of a five-person steering council. In January 2019, active Python core developers elected Brett Cannon, Nick Coghlan, Barry Warsaw, Carol Willing and Van Rossum to a five-member "Steering Council" to lead the project.

It is a multi-paradigm programming language. Object-oriented programming and structured programming are fully supported, and many of its features support functional programming and aspect-oriented programming (including by meta programming and metaobjects (magic methods)). Many other paradigms are supported via extensions, including design by contract and logic programming.

Python uses dynamic typing and a combination of reference counting and a cycle-detecting garbage collector for memory management. It also features dynamic name resolution (late binding), which binds method and variable names during program execution.

Python's design offers some support for functional programming in the Lisp tradition. It has filter, map, and reduce functions; list comprehensions, dictionaries, sets, and

generator expressions. The standard library has two modules (itertools and functools) that implement functional tools borrowed from Haskell and Standard ML and Data Science.

In 1962, John Tukey described a field he called “data analysis,” which resembles modern data science. Later, attendees at a 1992 statistics symposium at the University of Montpellier II acknowledged the emergence of a new discipline focused on data of various origins and forms, combining established concepts and principles of statistics and data analysis with computing.

A facial recognition system is a technology capable of matching a human face from a digital image or a video frame against a database of faces, typically employed to authenticate users through ID verification services, works by pinpointing and measuring facial features from a given image.

Development began on similar systems in the 1960s, beginning as a form of computer application. Since their inception, facial recognition systems have seen wider uses in recent times on smartphones and in other forms of technology, such as robotics. Because computerized facial recognition involves the measurement of a human's physiological characteristics, facial recognition systems are categorized as biometrics. Although the accuracy of facial recognition systems as a biometric technology is lower than iris recognition and fingerprint recognition, it is widely adopted due to its contactless process. Facial recognition systems have been deployed in advanced human-computer interaction, video surveillance and automatic indexing of images.

Facial recognition systems are employed throughout the world today by governments and private companies. Their effectiveness varies, and some systems have previously been scrapped because of their ineffectiveness. The use of facial recognition systems has also raised controversy, with claims that the systems violate citizens' privacy, commonly make incorrect identifications, encourage gender norms and racial profiling, and do not protect important biometric data. These claims have led to the ban of facial recognition systems in several cities in the United States.

Automated facial recognition was pioneered in the 1960s. Woody Bledsoe, Helen Chan Wolf, and Charles Bisson worked on using the computer to recognize human faces. Their early facial recognition project was dubbed "man-machine" because the coordinates of the facial features in a photograph had to be established by a human before they could be used by the computer for recognition. On a graphics tablet a human had to pinpoint the coordinates of facial features such as the pupil centres, the inside and outside corner of eyes, and the widows peak in the hairline. The coordinates were used to calculate 20 distances, including the width of the mouth and of the eyes. A human could process about 40 pictures an hour in this manner and so build a database of the computed distances. A computer would then automatically compare the distances for each photograph, calculate the difference between the distances and return the closed records as a possible match.

A video based pointing method will first track the coloured fingertip in frames and later apply English OCR above drawn image. It gives a characteristic human to machine connection in such a way that keypad is not required. Proposed methodology acquires the normal accuracy of over 92.083% and 50 ms delay per character. It can be used for all separated dialects but having red in the background can lead to false results. [1] Introduction of AR-Canvas for information visualization is beyond the traditional information visualization. Designing embedded visualization on an AR- Canvas requires revaluation design methods that we learned with traditional canvas. Main source of visual cluster may come from scenes with background and potentially hiding real world information and artifacts. [2]

In the approach of making documents in air utilizing Myo-armband, we divide the issue into two parts namely 2-DifViz and HDRA. The model accomplishes an acknowledgment of 91.7% on a freelance valuation and 96.7% on individual ward valuation. Visualization module 2-DiffViz produces multi-dimensional sensor information onto a 2D surface which produces a practical perception of the information. In identification module execute Handwritten Digits Recognition in Air (HDRA) the data represented by Myo-Armband is added into GRU (Gated Recurrent Unit) network. [3] Object character recognition is an efficient method of detecting the characters from the image. OCR is the is the electronic or mechanical transformation

of images of handwritten, printed, or typed text. It is mostly used in passport, documents, bank statements, invoices, business cards, computerized receipts, or any other suitable documents. OCR is a field of study in pattern recognition, computer vision and artificial intelligence. [4]

1.5 PROBLEM STATEMENT

To process facial data and train system to recognize and detect faces using numpy and open cv. Use numpy to convert images to numpy array.

1.6 OBJECTIVES

An application to train different facial data and train system to recognize and detect faces. Use numpy to convert facial data to a numpy array and store in a XML format.

CHAPTER 2

AIM & SCOPE OF FACE RECOGNITION

2.1 REQUIREMENTS

2.1.1 HARDWARE REQUIREMENTS

- Inter core i5 7th Gen / Ryzen 5 2500u
- 8GB of DDR4 RAM
- 128 GB SSD / HDD
- 2GB Nvidia GeForce RTX 1050 Ti
- Camera

2.1.2 SOFTWARE REQUIREMENTS:

- Python installed on your system
- Visual Studio Code

Visual Studio Code is an Integrated Development Environment (IDE) made by Microsoft for Windows, Linux and macOS. Features include support for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git. Users can change the theme, keyboard shortcuts, preferences, and install extensions that add additional functionality.

2.2 ROLE OF VISUAL STUDIO CODE

Visual Studio Code is a source-code editor that can be used with a variety of programming languages, including Java, JavaScript, Go, Node.js, Python and C++. It is based on the Electron framework, which is used to develop Node.js Web applications that run on the Blink layout engine. Visual Studio Code employs the same editor component (codenamed "Monaco") used in Azure DevOps (formerly called Visual Studio Online and Visual Studio Team Services).

Instead of a project system, it allows users to open one or more directories, which can then be saved in workspaces for future reuse. This allows it to operate as a language-agnostic code editor for any language. It supports a number of programming languages and a set of features that differs per language. Unwanted files and folders can be excluded from the project tree via the settings. Many Visual Studio Code features are not exposed through menus or the user interface but can be accessed via the command palette.

Visual Studio Code can be extended via extensions, available through a central repository. This includes additions to the editor and language support. A notable feature is the ability to create extensions that add support for new languages, themes, and debuggers, perform static code analysis, and add code linters using the Language Server Protocol.

Visual Studio Code includes multiple extensions for FTP, allowing the software to be used as a free alternative for web development. Code can be synced between the editor and the server, without downloading any extra software.

Visual Studio Code allows users to set the code page in which the active document is saved, the newline character, and the programming language of the active document. This allows it to be used on any platform, in any locale, and for any given programming language.

2.2.1 LANGUAGES USED FOR PROJECT

Machine Learning , data science and artificial Intelligence-based projects are obviously what the future holds . We want better personalization, smarter recommendations, and

improved search functionality. Our apps can see, hear, and respond-that's what artificial intelligence (AI) has brought , enhancing the user experience and creating value across many industries.

FEATURES OF PYTHON:

Following are some of the important features that make Python the first choice of Data Science Engineers.

Simple and consistent:

Python offers concise and readable code. While complex algorithms and versatile workflows stand behind data science, machine learning and AI, Python's simplicity allows developers to write reliable systems. Python code is understandable by humans, which makes it easier to build models for machine learning.

Extensive selection of libraries and frameworks:

Implementing Data Science and ML algorithms can be tricky and requires a lot of time. It's vital to have a well-structured and well tested environment to enable developers to come up with the best coding solutions.

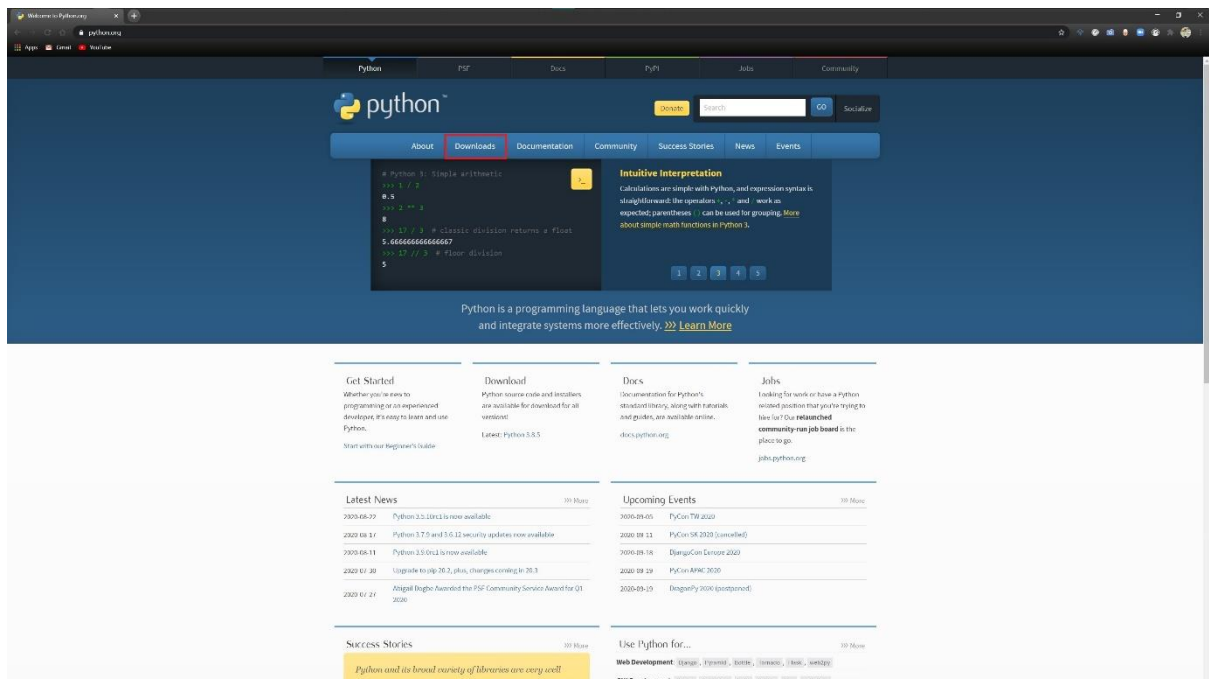
- Numpy and Pandas for storing images in an array.
- Open CV for detecting dace.
- Tkinter for graphical user interface.

Platform independence:

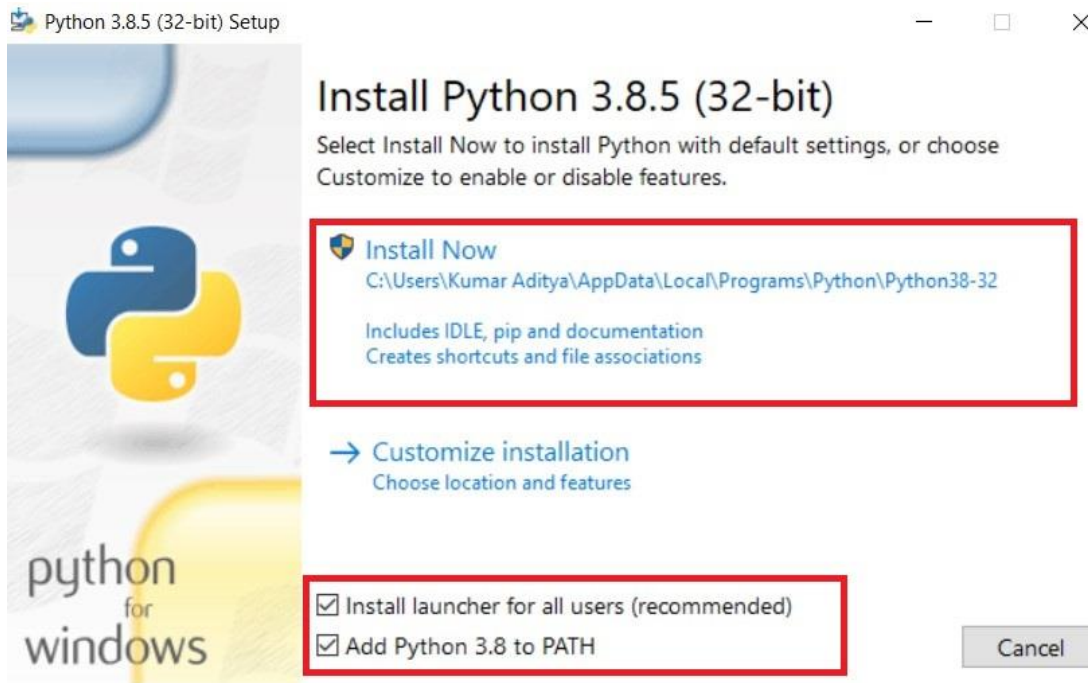
Python software can be easily distributed and used on that operating system without a Python interpreter. Great Community and Popularity: In the Developer survey 2018 by Stack Overflow, Python was among the most popular programming language, which ultimately mean that you can find easily a development company.

2.3 HOW TO INSTALL PYTHON

From Website:



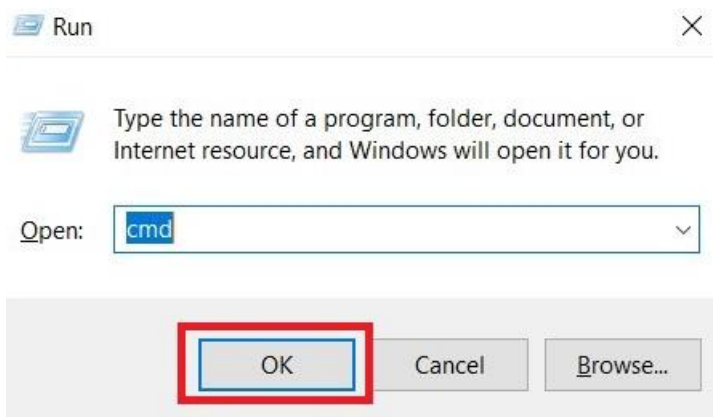
You can download Python from their website directly to your system for development purposes.



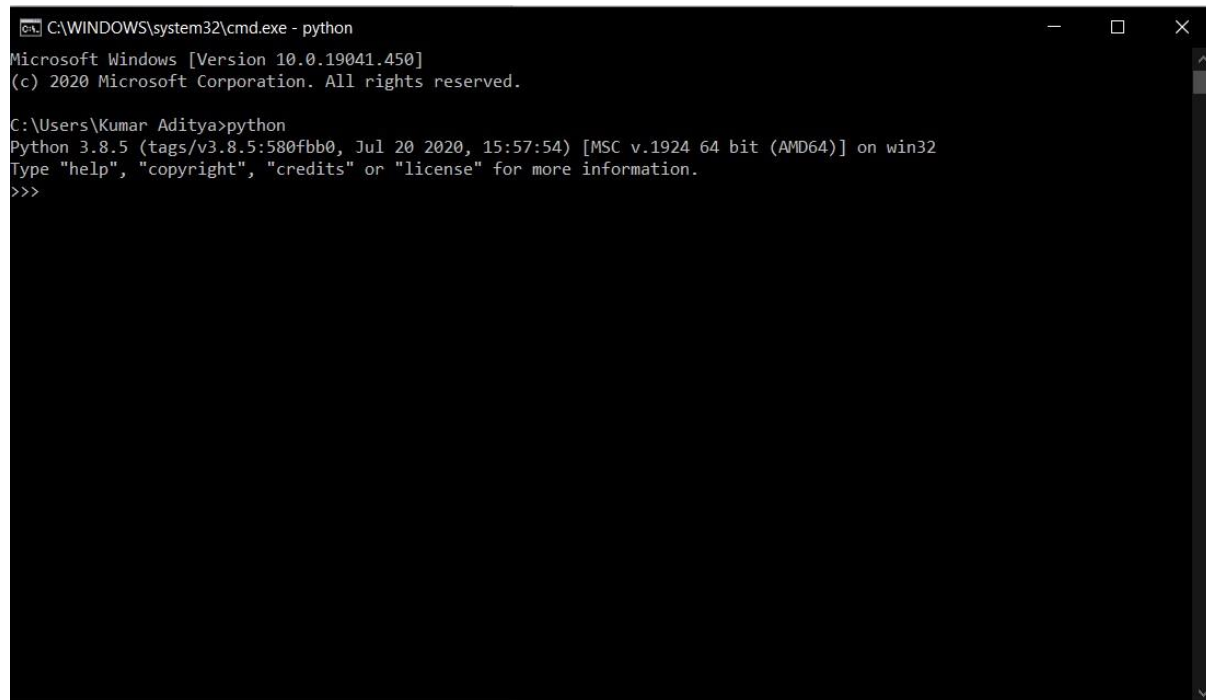
Do not forget to “Add Python to PATH”

- ❖ After installation is complete you will be able to see Python in Command Prompt.

Run -> cmd



- ❖ You can see information about your version and switch to python by typing “python” in Command Prompt.



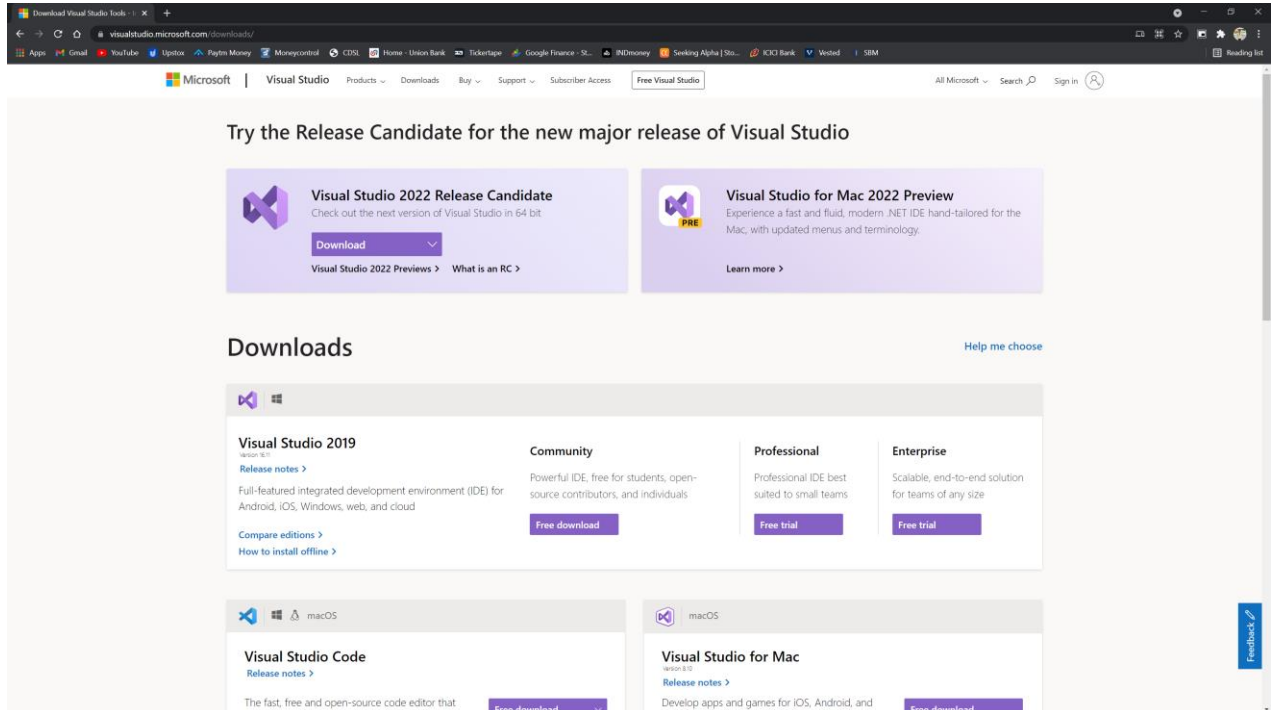
```
C:\WINDOWS\system32\cmd.exe - python
Microsoft Windows [Version 10.0.19041.450]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\Kumar Aditya>python
Python 3.8.5 (tags/v3.8.5:580fbb0, Jul 20 2020, 15:57:54) [MSC v.1924 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

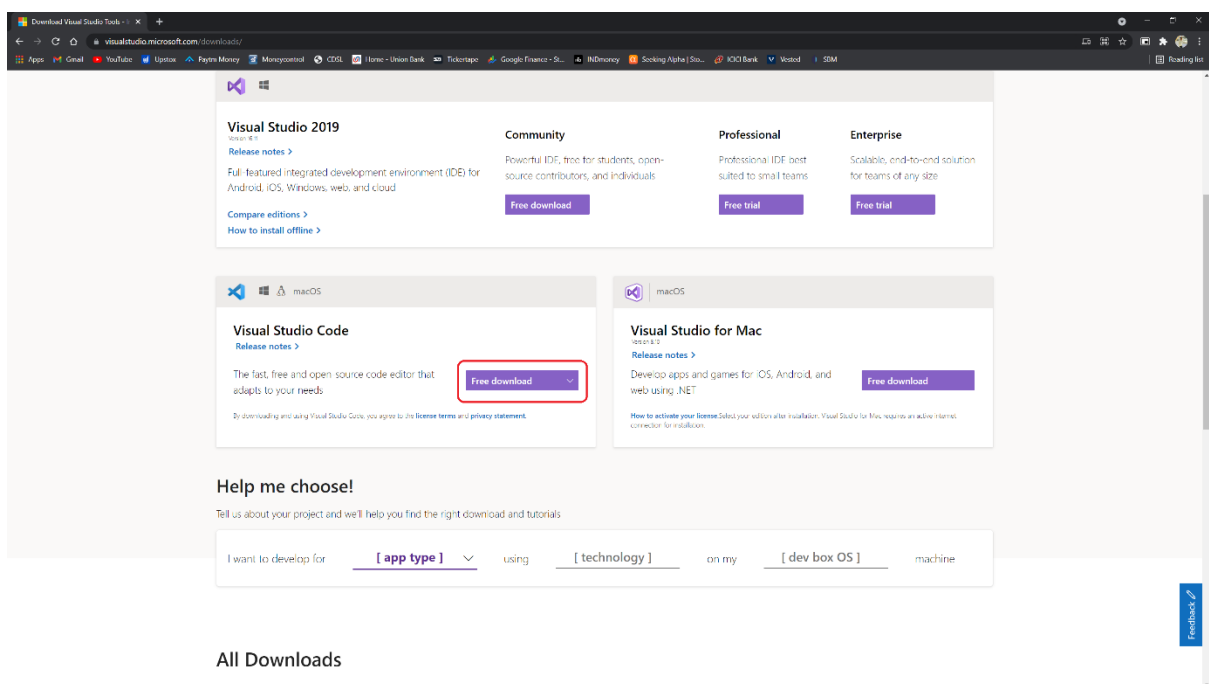
2.4 HOW TO INSTALL VISUAL STUDIO

Go to website:

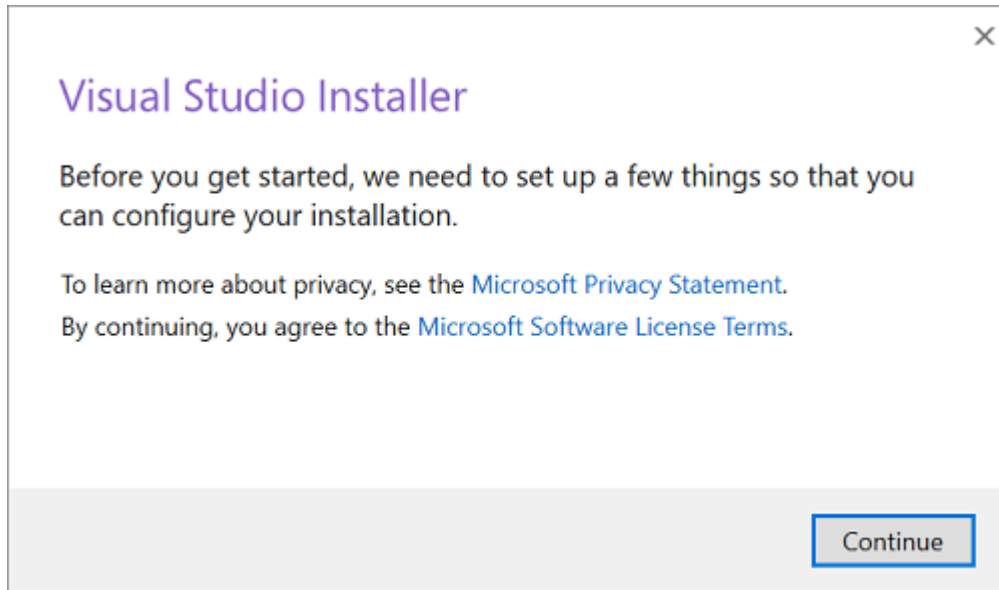
<https://visualstudio.microsoft.com/downloads/>



Use the download link:

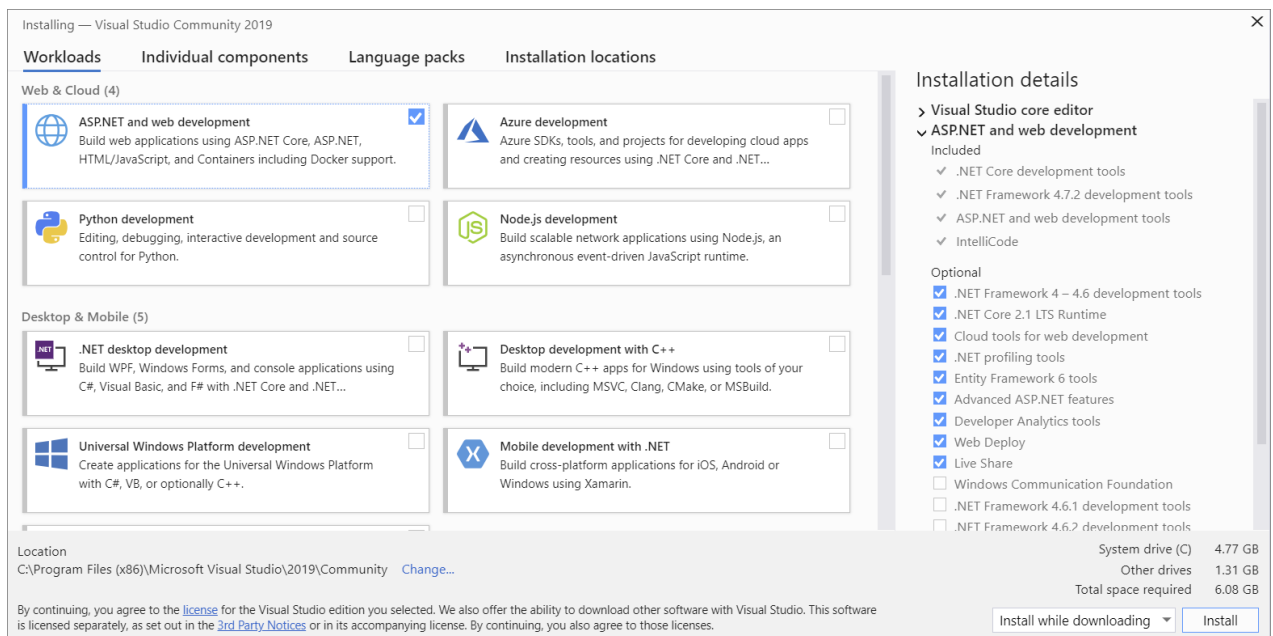


Install the Visual Studio Installer:



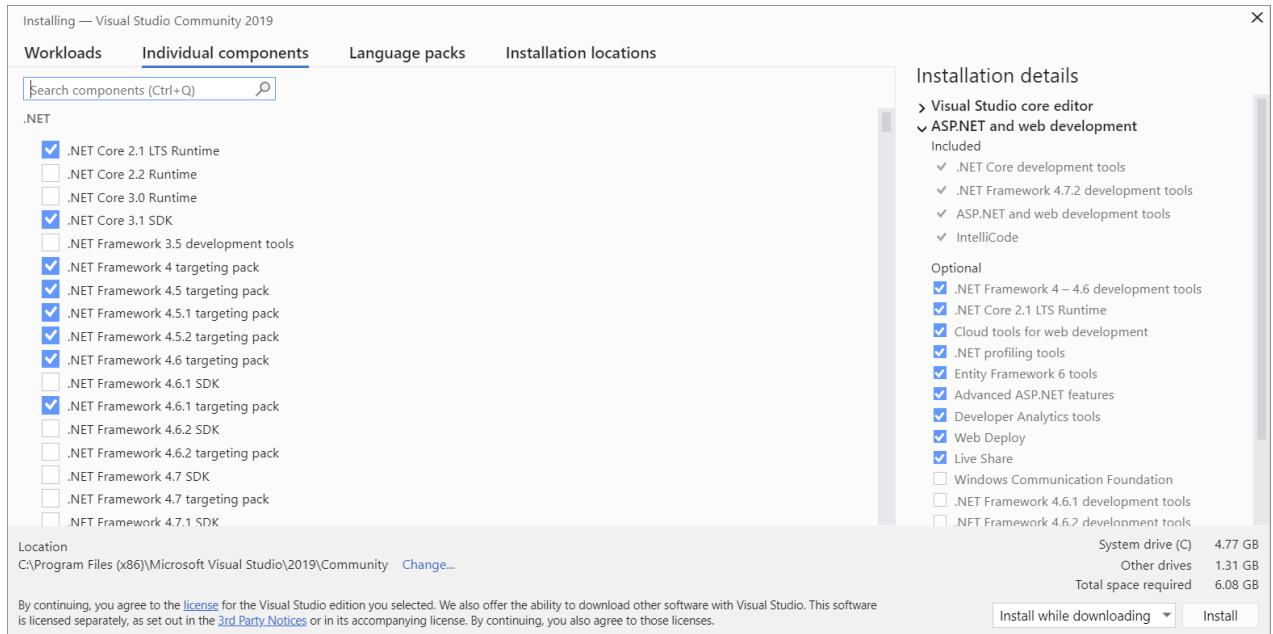
Choose workloads:

After the installer is installed, you can use it to customize your installation by selecting the feature set or workloads that you want.



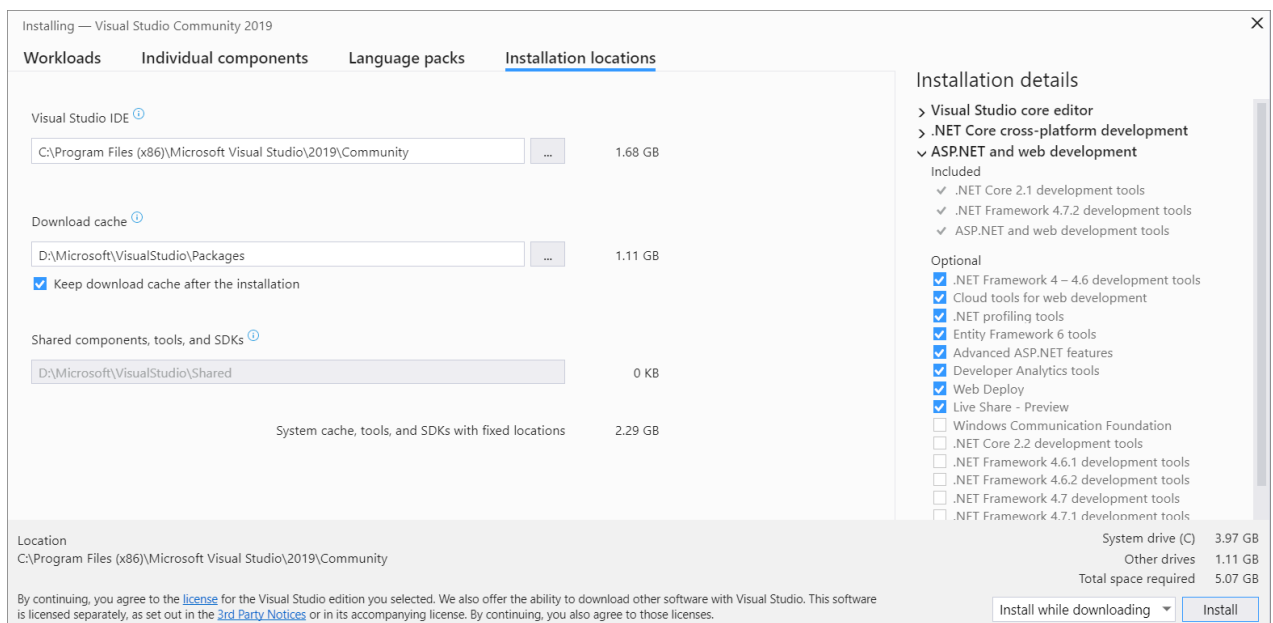
Choose individual components (optional):

If you don't want to use the Workloads feature to customize your Visual Studio installation, or you want to add more components than a workload installs, you can do so by installing or adding individual components from the Individual components tab.



Select the installation location (optional):

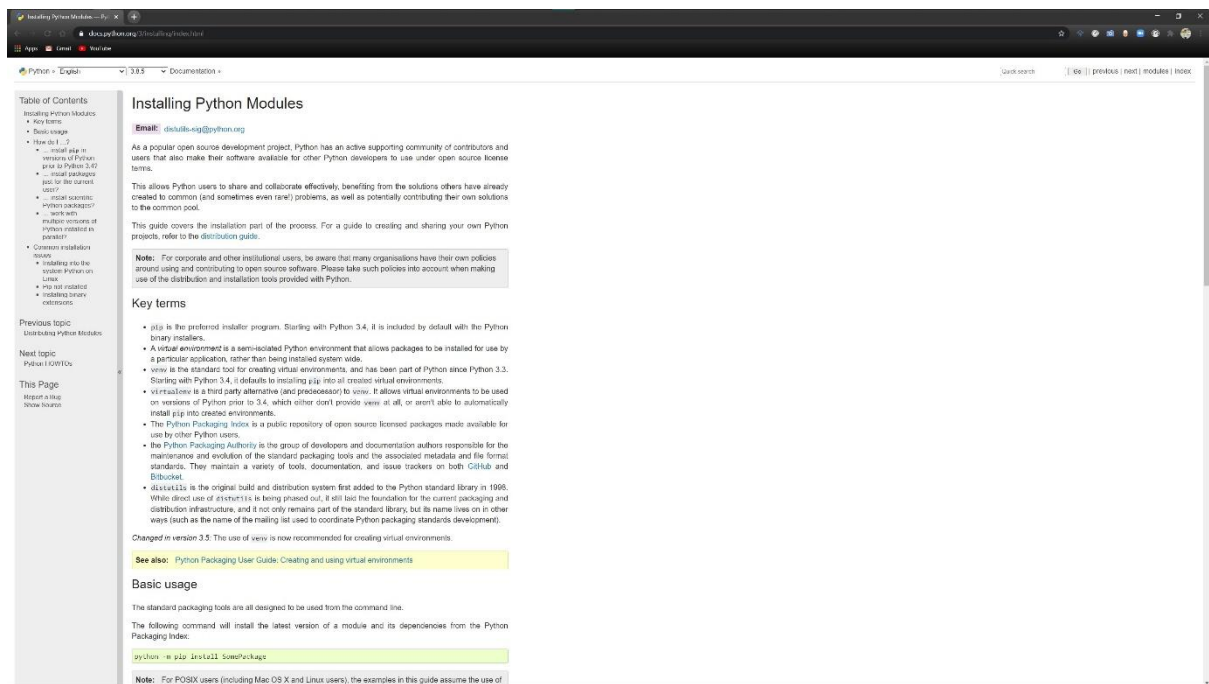
You can reduce the installation footprint of Visual Studio on your system drive. You can choose to move the download cache, shared components, SDKs, and tools to different drives, and keep Visual Studio on the drive that runs it the fastest.



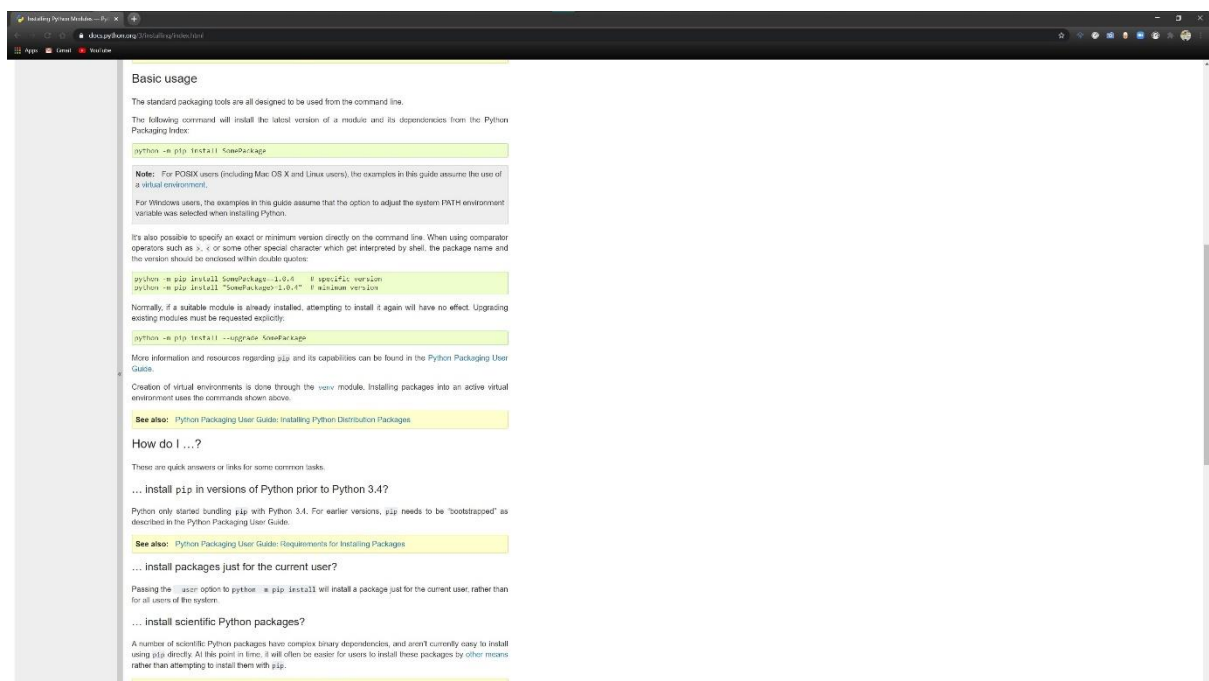
Start developing:

1. After Visual Studio installation is complete, select the Launch button to get started developing with Visual Studio.
2. On the start window, choose Create a new project.
3. In the search box, enter the type of app you want to create to see a list of available templates. The list of templates depends on the workloads that you chose during installation. To see different templates, choose different workloads.
4. You can also filter your search for a specific programming language by using the Language drop-down list. You can filter by using the Platform list and the Project type list, too.
5. Visual Studio opens your new project, and you're ready to code!

2.5 INSTALLING PYTHON MODULES:



python -m pip install SomePackage



CHAPTER 3

WHY OBJECT TRACKING

Object tracking is an application of deep learning where the program takes an initial set of object detections and develops a unique identification for each of the initial detections and then tracks the detected objects as they move around frames in a video.

In other words, object tracking is the task of automatically identifying objects in a video and interpreting them as a set of trajectories with high accuracy.

Often, there's an indication around the object being tracked, for example, a surrounding square that follows the object, showing the user where the object is on the screen

3.1 Uses and Types of Object Tracking

Object tracking is used for a variety of use cases involving different types of input footage. Whether or not the anticipated input will be an image or a video, or a real-time video vs. a prerecorded video, impacts the algorithms used for creating object tracking applications.

The kind of input also impacts the category, use cases, and applications of object tracking. Here, we will briefly describe a few popular uses and types of object tracking, such as video tracking, visual tracking, and image tracking.

3.1.1 Video Tracking

Video tracking is an application of object tracking where moving objects are located within video information. Hence, video tracking systems are able to process live, real-time footage and also recorded video files.

The processes used to execute video tracking tasks differ based on which type of video input is targeted. This will be discussed more in-depth when we compare batch and online tracking methods later in this article.

Different video tracking applications play an important role in video analytics, in scene understanding for security, military, transportation, and other industries. Today, a

wide range of real-time computer vision and deep learning applications use videotracking methods. I recommend you check out our extensive list of the most popular Computer Vision Applications in 2021

3.1.2 Visual Tracking

Visual tracking or visual target-tracking is a research topic in computer vision that is applied in a large range of everyday scenarios. The goal of visual tracking is to estimate the future position of a visual target that was initialized without the availability of the rest of the video.

3.1.3 Image Tracking

Image tracking is meant for detecting two-dimensional images of interest in a given input. That image is then continuously tracked as they move in the setting.

Image tracking is ideal for datasets with highly contrasting images (ex. black and white), asymmetry, few patterns, and multiple identifiable differences between the image of interest and other images in the image set.

Image tracking relies on computer vision to detect and augment images after image targets are predetermined.

3.1.4 Object tracking camera

Modern object tracking methods can be applied to real-time video streams of basically any camera. Therefore, the video feed of a USB camera or an IP camera can be used to perform object tracking, by feeding the individual frames to a tracking algorithm. Frame skipping or parallelized processing are common methods to improve object tracking performance with real-time video feeds of one or multiple cameras.

3.2 What makes Object Tracking difficult? What are the common challenges and advantages of Object Tracking?

The main challenges usually stem from issues in the image that make it difficult for object tracking models to effectively perform detections on the images.

Here, we will discuss the few most common issues with the task of tracking objects and methods of preventing or dealing with these challenges.

3.2.1. Training and Tracking Speed

Algorithms for tracking objects are supposed to not only accurately perform detections and localize objects of interest but also do so in the least amount of time possible. Enhancing tracking speed is especially imperative for real-time object tracking models.

To manage the time taken for a model to perform, the algorithm used to create the object tracking model needs to be either customized or chosen carefully. Fast R-CNN and Faster R-CNN can be used to increase the speed of the most common R-CNN approach.

Since CNNs (Convolutional Neural Networks) are commonly used for object detection, CNN modifications can be the differentiating factor between a faster object tracking model and a slower one. Design choices besides the detection framework also influence the balance between speed and accuracy of an object detection model.

3.2.2. Background Distractions

The backgrounds of inputted images or images used to train object tracking models also impact the accuracy of the model. Busy backgrounds of objects meant to be tracked can make it harder for small objects to be detected.

With a blurry or single color background, it is easier for an AI system to detect and track objects. Backgrounds that are too busy, have the same color as the object, or that are too cluttered can make it hard to track results for a small object or a lightly colored object.

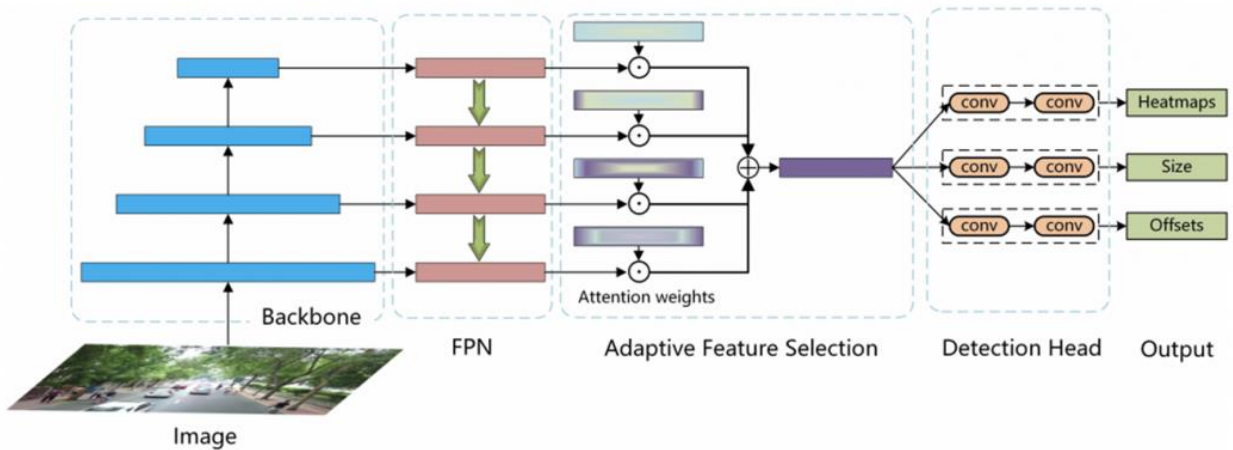
3.2.3. Multiple Spatial Scales

Objects meant to be tracked can come in a variety of sizes and aspect ratios. These ratios can confuse the object tracking algorithms into believing objects are scaled larger or smaller than their actual size. The size misconceptions can negatively impact detections or detection speed.

To combat the issue of varying spatial scales, programmers can implement techniques such as feature maps, anchor boxes, image pyramids, and feature pyramids.

- **Anchor Boxes:** Anchor boxes are a compilation of bounding boxes that have a specified height and width. The boxes are meant to acquire the scale and aspect ratios of objects of interest. They are chosen based on the average object size of the objects in a given dataset. Anchor boxes allow various types of objects to be detected without having the bounding box coordinates alternated during localization.
- **Feature Maps:** A feature map is the output image of a layer when a Convolutional Neural Network (CNN) is used to capture the result of applying filters to that input image. Feature maps allow a deeper understanding of the features being detected by a CNN. Single-shot detectors have to take into account the issue of multiple scales because they detect objects with just one pass through a CNN framework. This will occur in a detection decrease for small images. Small images can lose signal during downsampling in the pooling layers, which is when the CNN was trained on a low subset of those smaller images. Even if the number of objects is the same, downsampling can occur because the CNN wasn't able to detect the small images and count them towards the sample size. To prevent this, multiple feature maps can be used to allow single-shot detectors to look for objects within CNN layers – including earlier layers with higher resolution images. Single-shot detectors (SSD) are still not an ideal option for small object tracking because of the difficulty they experience when detecting small objects. Tight groupings can prove especially difficult. For instance, overhead drone shots of a group of herd animals will be difficult to track using single-shot detectors.
- **Image and Feature Pyramid Representations:** Feature pyramids, also known as multi-level feature maps because of their pyramidal structure, are a preliminary solution for object scale variation when using object tracking datasets. Hence, feature pyramids model the most useful information regarding objects of different sizes in a top-down representation and therefore make it

easier to detect objects of varying sizes. Strategies such as image pyramids and feature pyramids are useful for preventing scaling issues. The feature pyramid is based on multi-scale feature maps, which uses less computational energy than image pyramids. This is because image pyramids consist of a set of resized versions of one input image that are then sent to the detector at testing.



Occlusion has a lot of definitions. In medicine, occlusion is the “blockage of a blood vessel” due to the vessel merging to a close; in deep learning, it has a similar meaning.

In AI vision tasks using deep learning, occlusion happens when multiple objects come too close together (merge). This causes issues for object tracking systems because the occluded objects are seen as one or simply track the object incorrectly. The system can get confused and identify the initially tracked object as a new object.

Occlusion sensitivity prevents this misidentification by allowing the user to understand which parts of an image are the most important for the object tracking system to classify. Occlusion sensitivity refers to a measure of the network’s sensitivity to occlusion in different data regions. It is done using small subsets of the original dataset.

Levels of Object Tracking

Object Tracking consists of multiple subtypes because it is such a broad application. Levels of object tracking differ depending on the number of objects being tracked.

Multiple Object Tracking (MOT)

Multiple object tracking is defined as the problem of automatically identifying multiple objects in a video and representing them as a set of trajectories with high accuracy.

Hence, multi-object tracking aims to track more than one object in digital images. It is also called multi-target tracking, as it attempts to analyze videos to identify objects (“targets”) that belong to more than one predetermined class

Multiple object tracking is of great importance in autonomous driving, where it is used to detect and predict the behavior of pedestrians or other vehicles. Hence, the algorithms are often benchmarked on the KITTI tracking test. KITTI is a challenging real-world computer vision benchmark and image dataset, popularly used in autonomous driving.

In 2022, the best performing multiple object tracking algorithms are DEFT (88.95 MOTA, Multiple Object Tracking Accuracy), CenterTrack (89.44 MOTA), and SRK ODESA (90.03 MOTA).

Multiple Object Tracking (MOT) vs. General Object Detection

Object detections typically produce a collection of bounding boxes as outputs. Multiple object tracking often has little to no prior training regarding the appearance and number of targets. Bounding boxes are identified using their height, width, coordinates, and other parameters. Meanwhile, MOT algorithms additionally assign a target ID to each bounding box. This target ID is known as a detection, and it is important because it allows the model to distinguish among objects within a class.

For example, instead of identifying all cars in a photo where multiple cars are present as just “car,” MOT algorithms attempt to identify different cars as being different from each other rather than all falling under the “car” label.

Single Object Tracking

Single Object Tracking (SOT) creates bounding boxes that are given to the tracker based on the first frame of the input image. Single Object Tracking is also sometimes known as Visual Object Tracking.

SOT implies that one singular object is tracked, even in environments involving other objects. Single Object Trackers are meant to focus on one given object rather than multiple. The object of interest is determined in the first frame, which is where the object to be tracked is initialized for the first time. The tracker is then tasked with locating that unique target in all other given frames.

SOT falls under the detection-free tracking category, which means that it requires manual initialization of a fixed number of objects in the first frame. These objects are then localized in consequent frames. A drawback of detection-free tracking is that it cannot deal with scenarios where new objects appear in the middle frames. SOT models should be able to track any given object.

Object Tracking Algorithms

Multiple Object Tracking (MOT) Algorithm Introduction

Most multiple object tracking algorithms incorporate an approach called tracking-by-detection. The tracking-by-detection method involves an independent detector that is applied to all image frames to obtain likely detections, and then a tracker, which is run on the set of detections. Hereby, the tracker attempts to perform data association (for example, linking the detections to obtain complete trajectories). The detections extracted from video inputs are used to guide the tracking process by connecting them and assigning identical IDs to bounding boxes containing the same target.

- **Batch method:** Batch tracking algorithms use information from future video frames when deducing the identity of an object in a certain frame. Batch tracking algorithms use non-local information regarding the object. This methodology results in a better quality of tracking.
- **Online method:** While batch tracking algorithms access future frames, online tracking algorithms only use present and past information to come to conclusions regarding a certain frame.

Online tracking methods for performing MOT generally perform worse than batch methods because of the limitation of online methods staying constrained to the present frame. However, this methodology is sometimes necessary because of the use case.

For example, real-time problems requiring the tracking of objects, like navigation or autonomous driving, do not have access to future video frames, which is why online tracking methods are still a viable option.

Multiple Object Tracking Algorithm Stages

Most multiple object tracking algorithms contain a basic set of steps that remain constant as algorithms vary. Most of the so-called multi-target tracking algorithms share the following stages:

- **Stage #1: Designation or Detection:** Targets of interest are noted and highlighted in the designation phase. The algorithm analyzes input frames to identify objects that belong to target classes. Bounding boxes are used to perform detections as part of the algorithm.
- **Stage #2: Motion:** Feature extraction algorithms analyze detections to extract appearance and interaction features. A motion predictor, in most cases, is used to predict subsequent positions of each tracked target.
- **Stage #3: Recall:** Feature predictions are used to calculate similarity scores between detection couplets. Those scores are then used to associate detections that belong to the same target. IDs are assigned to similar detections, and different IDs are applied to detections that are not part of pairs.

Some object tracking models are created using these steps separately from each other, while others combine and use the steps in conjunction. These differences in algorithm processing create unique models where some are more accurate than others.

Popular Object Tracking Algorithms

Convolutional Neural Networks (CNN) remain the most used and reliable network for object tracking. However, multiple architectures and algorithms are being explored as well. Among these algorithms are Recurrent Neural Networks (RNNs), Autoencoders (AEs), Generative Adversarial Networks (GANs), Siamese Neural Networks (SNNs), and custom neural networks.

Although object detectors can be used to track objects if it is applied frame-by-frame, this is a computationally limiting and therefore a rather inefficient method of performing object tracking. Instead, object detection should be applied once, and then the object tracker can handle every frame after the first. This is a more computationally effective and less cumbersome process of performing object tracking.

1. OpenCV Object Tracking

OpenCV object tracking is a popular method because OpenCV has so many algorithms built-in that are specifically optimized for the needs and objectives of object or motion tracking.

Specific Open CV object trackers include the BOOSTING, MIL, KCF, CSRT, MedianFlow, TLD, MOSSE, and GOTURN trackers. Each of these trackers is best for different goals. For example, CSRT is best when the user requires a higher object tracking accuracy and can tolerate slower FPS throughput.

The selection of an OpenCV object tracking algorithm depends on the advantages and disadvantages of that specific tracker and the benefits:

- The KCF tracker is not as accurate compared to the CSRT but provides comparably higher FPS.
- The MOSSE tracker is very fast, but its accuracy is even lower than tracking with KCF. Still, if you are looking for the fastest object tracking OpenCV method, MOSSE is a good choice.
- The GOTURN tracker is the only detector for deep learning based object tracking with OpenCV. The original implementation of GOTURN is in Caffe, but it has been ported to the OpenCV Tracking API.

2. DeepSORT

DeepSORT is a good object tracking algorithm choice, and it is one of the most widely used object tracking frameworks. Appearance information is integrated within the algorithm, which vastly improves DeepSORT performance. Because of the

integration, objects are trackable through longer periods of occlusion – reducing the number of identity switches.

For complete information on the inner workings of DeepSORT and specific algorithmic differences between DeepSORT and other algorithms, we suggest the article “Object Tracking using DeepSORT in TensorFlow 2” by Anushka Dhiman.

3. Object Tracking MATLAB

MATLAB is a numeric computing platform, which makes it different in its implementation compared to DeepSORT and OpenCV, but it is nevertheless a fine choice for visual tracking tasks.

The Computer Vision Toolbox in MATLAB provides video tracking algorithms, such as continuously adaptive mean shift (CAMShift) and Kanade-Lucas-Tomasi (KLT) for tracking a single object or for use as building blocks in a more complex tracking system.

4. MDNet

MDNet is a fast and accurate, CNN-based visual tracking algorithm inspired by the R-CNN object detection network. It functions by sampling candidate regions and passing them through a CNN. The CNN is typically pre-trained on a vast dataset and refined at the first frame in an input video.

Therefore, MDNet is most useful for real-time object tracking use cases. However, while it suffers from high computational complexity in terms of speed and space, it still is an accurate option.

The computation-heavy aspects of MDNet can be minimized by performing RoI (Region of Interest) Pooling, however, which is a relatively effective way of avoiding repetitive observations and accelerating inference.

CHAPTER 4

METHODS AND MATERIAL USED

3.1 MATERIAL USED

- Camera
- Visual Studio Code

3.2 METHODS

3.2.1 DESIGNING THE PROJECT

Face detection is a computer vision technology that helps to locate/visualize human faces in digital images. This technique is a specific use case of object detection technology that deals with detecting instances of semantic objects of a certain class (such as humans, buildings or cars) in digital images and videos. With the advent of technology, face detection has gained a lot of importance especially in fields like photography, security, and marketing.

Images as Arrays

An image is nothing but a standard Numpy array containing pixels of data points. More

the number of pixels in an image, the better is its resolution. You can think of pixels to be tiny blocks of information arranged in the form of a 2 D grid, and the depth of a pixel refers to the color information present in it. In order to be processed by a computer, an image needs to be converted into a binary form.

Naturally, more the number of bits/pixels, more possible colors in the images. The following table shows the relationship more clearly.

Bits/Pixel	Possible colours
1	$2^1 = 2$
2	$2^2 = 4$
3	$2^3 = 8$
4	$2^4 = 16$
8	$2^8 = 256$
16	$2^{16} = 65000$

Table 3.1 Possible colour combination

Grayscale image

A grayscale image consists of 8 bits per pixel. This means it can have 256 different shades where 0 pixels will represent black color while 255 denotes white. For example, the image below shows a grayscale image represented in the form of an array. A grayscale image has only 1 channel where the channel represents dimension.

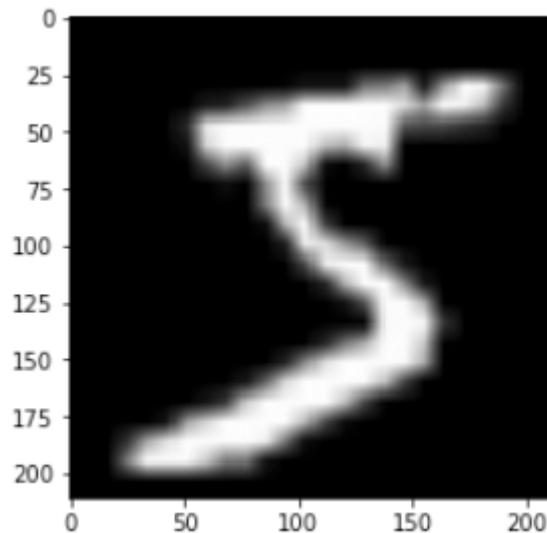


Fig 3.1 Grayscale Image

Colored image

Colored images are represented as a combination of Red, Blue, and Green, and all the other colors can be achieved by mixing these primary colors in correct proportions.

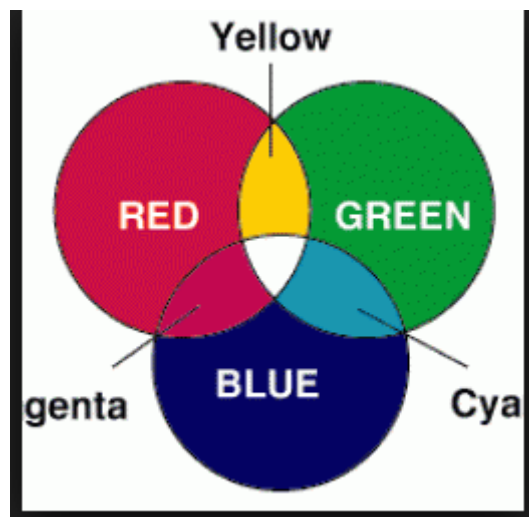


Fig 3.2 Colored Image

A colored image also consists of 8 bits per pixel. As a result, 256 different shades of colors can be represented with 0 denoting black and 255 white. Let us look at the famous colored image of a mandrill which has been cited in many image processing examples

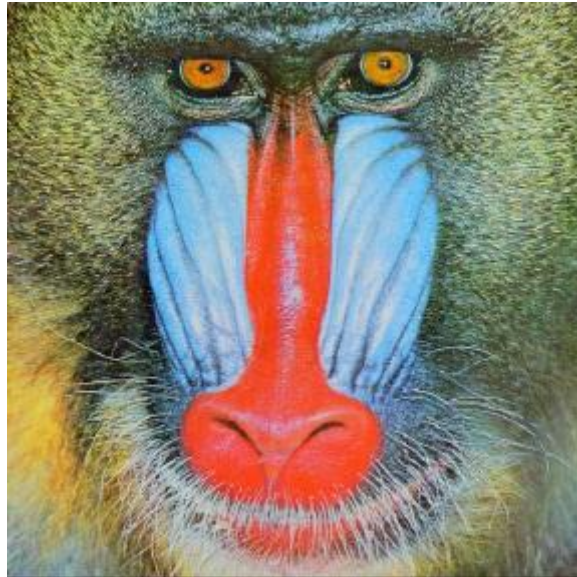


Fig 3.3 Representation of colored image

Face detection is a technique that identifies or locates human faces in digital images. A typical example of face detection occurs when we take photographs through our smartphones, and it instantly detects faces in the picture. Face detection is different from Face recognition. Face detection detects merely the presence of faces in an image while facial recognition involves identifying whose face it is. In this article, we shall only be dealing with the former.

Face detection is performed by using classifiers. A classifier is essentially an algorithm that decides whether a given image is positive(face) or negative(not a face). A classifier needs to be trained on thousands of images with and without faces. Fortunately, OpenCV already has two pre-trained face detection classifiers, which can readily be used in a program. The two classifiers are:

- Haar Classifier and
- Local Binary Pattern(LBP) classifier.

Load the necessary Libraries:

```
import numpy as np
import cv2
import matplotlib.pyplot as plt
```

Loading the image to be tested in grayscale:

```
#Loading the image to be tested
test_image = cv2.imread('data/baby1.jpg')

#Converting to grayscale
test_image_gray = cv2.cvtColor(test_image, cv2.COLOR_BGR2GRAY)

# Displaying the grayscale image
plt.imshow(test_image_gray, cmap='gray')
```

Haar Cascade files:

```
haar_cascade_face = cv2.CascadeClassifier('data/haarcascade/haarcascade_frontalface_default.xml')
```

Face Detection:

- **scalefactor** In a group photo, there may be some faces which are near the camera than others. Naturally, such faces would appear more prominent than the ones behind. This factor compensates for that.
- **minNeighbors** This parameter specifies the number of neighbors a rectangle should have to be called a face. You can read more about it.

```
faces_rects = haar_cascade_face.detectMultiScale(test_image_gray, scaleFactor = 1.2, minNeighbors =

# Let us print the no. of faces found
print('Faces found: ', len(faces_rects))

Faces found: 1
```

CHAPTER 5

RESULT AND DISCUSSION

RESULT

Now we have connected everything together and now we will try to train data by calling the functions.



Fig 4.1 Training Data

Final Result:

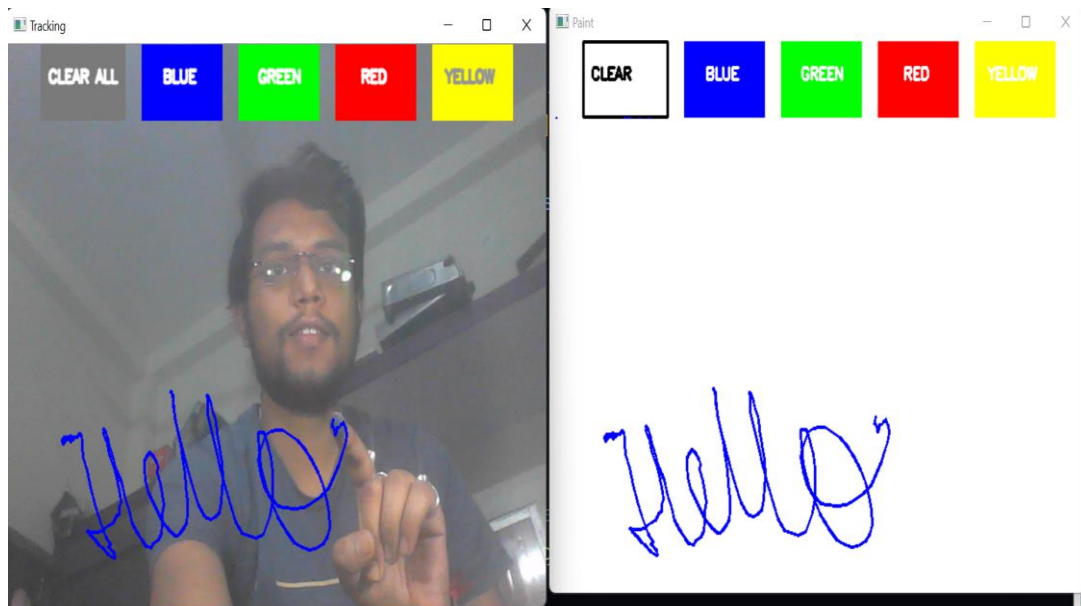


Fig 4.3 Final Result

DISCUSSION

New sensory innovation like these takes into account for free air sensing at minimal price. Over the long run, these interactions will increment accuracy, empowering the sensation of individual hands, fingers, and different items. Embedding data visibility in the real world around us has recently become more entrenched as a single visual field with potentially comprehensive applications. AOT (Automatic object tracking) is being used in many fields such as computer vision and human to machine association. Often, an object can be a text or an individual who had to be traced. In the composition, an alternate use of the tracking algorithm is presented. Specialist used to translate Sign Languages, some used it to recognize sign language.

A real time fast video-based fingertip tracing and perceiving algorithm is introduced. The proposed algorithm has two main functions: first detecting coloured finger movement in video grouping and afterward utilizing OCR (Optical Character Reorganization) to visualize the edited image. The proposed strategy is a software-based methodology. Instead of using sensory-powered portable devices, such as the

iPhone or Wii mote, we use a different Zero Touch sensor access to allow visual, integrated, and remote-control canvas.

CHAPTER 6

CONCLUSION & FUTURE WORK

SUMMARY:

This application can be an excellent option for smart wearables and Augmented reality implementation. Colorful lines can be drawn on the desire of the user. It can serve as a tool in helping especially abled people.

FUTURE WORK:

This work can further be improved by adding multicore module. Furthermore, we can use different methods such as pygame for drawing lines. Implementation of different brush types, shapes and textures can also give this tool a sturdy look as a drawing application. We can use hand gesture technique to future use it as an air writing application. The system can also be implemented with playback animation module. Implementation of algorithm like YOLO can further improve the accuracy of the system.

CONCLUSION:

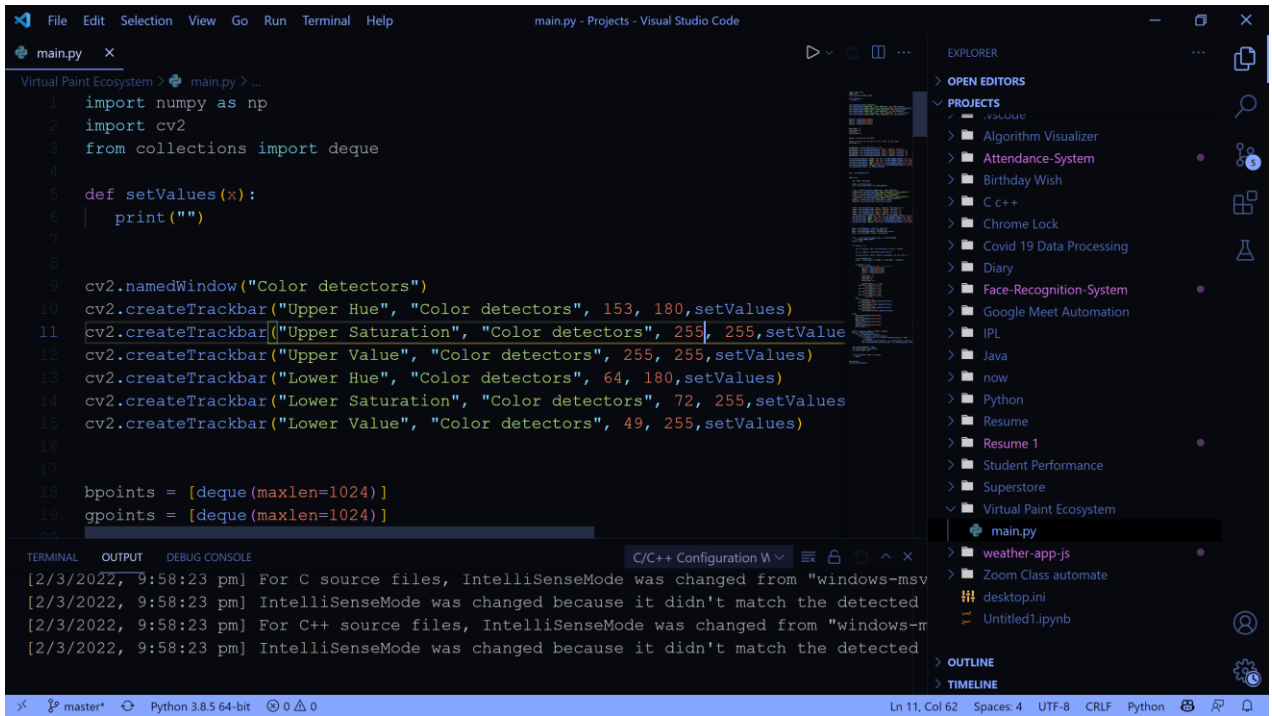
The system has potential to easily challenge traditional drawing methods. It removes the need to carry a stylus or any other hardware. This application can be an excellent option for smart wearables and Augmented reality implementation. Colorful lines can be drawn on the desire of the user. It can serve as a tool in helping especially abled people.

REFERENCES

- [1] "Liu W, Chen X, Lei X, Rian H, & Wang Y" (2017). Surface processing of polyester canvas using atmospheric pressure air glow discharge plasma. *Plasma Chemistry and Plasma Processing*, 37(2), 465-474. J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.
- [2] "Moeller, I., Lunfer, N., Hamilton, R., Lin, H., & Kerne, A." (2011). IntangibleCanvas: free-air finger painting on a projected canvas. In *CHI'11 Extended Abstracts on Human Factors in Computing Systems* (pp. 1615-1620).
- [3] "Hackney, S. (2004)". Paintings on canvas: lining and alternatives. *Tate papers*, 2.
- [4] "Rach, R., Sicat, R., Pfister, H., & Quigley", A. (2017). Drawing into the AR-CANVAS: Designing embedded
- [5] "Saini, S. H., Dua, N., Choudhary, A. K., & Phogat, R." (2021). AIR CANVAS APPLICATION USING OPENCV AND NUMPY IN PYTHON.
- [6] "Kalair, A. R., Dilshad, S., Ahas, N., Sevedmahmoudian, M., Stojanovski, A., & Koh, K." (2021). Application of Business Model Canvas for Solar Thermal Air Conditioners. *Front. Energy Res*, 9, 671973.
- [7] "Ulrich, M., Klemm, M., Plehner, K. O., & Hornung, M." (2018). Airline categorisation by applying the business model canvas and clustering algorithms. *Journal of Air Transport Management*, 71, 175-192.
- [8] "Adam, A. I., Planken, P. C., Meloni, S., & Dik, I." (2000). Terahertz imaging of hidden paint layers on canvas. *Optics Express*, 17(5), 3407-3416.
- [9] Gianni, P., Dei, I., Ceccato, M., Schettino, C., & Baglini, P. (2002). Nanotechnologies for conservation of cultural heritage: paper and canvas deacidification. *Langmuir*, 18(21), 8198-8203.
- [10] "Zhu, G. Q., Zhang, G. W., & Huang, L. L." (2013). Experimental study of upward flame spread over canvas hung in the air. *Journal of applied fire science*, 23(3), 349-363.
- [11] "Seves, A. M., Sora, S., Scicolone, G., Testa, G., Bonfatti, A. M., Rossi, F., & Seves, A." (2000). Effect of thermal accelerated ageing on the properties of model canvas paintings. *Journal of Cultural heritage*, 1(3), 315-322.
- [12] Dixon, T. (2020). Framing, glazing, backing and hanging of paintings on canvas. In *Conservation of easel paintings* (pp. 748-753). Routledge.
- [13] Schuiling, J. H. N. (2020). Booking and Payment Integration of Air and Public Transport for First/Last Mile Airport Trips: A Conceptual Framework and Business Model Canvas.
- [14] López-Miras, M., Piñar, G., Romero-Noguera, I., Bolívar-Galiano, E. C., Etteneauer, I., Sterflinger, K., & Martín-Sánchez, I. (2013). Microbial communities adhering to the obverse and reverse sides of an oil painting on canvas: identification and evaluation of their biodegradative potential. *Aerobiologia*, 29(2), 301-314.
- [15] "Zhang, H., Sfarra, S., Osman, A., Ibarra-Castaneda, C., Theodorakeas, P., Avdelidis, N. P., & Maldague, X. P." (2017, September). Nondestructive investigation of paintings on canvas by infrared thermography, air-coupled ultrasound, and X-ray radiography. In *International Symposium on the Conservation of Monuments in the Mediterranean Basin* (pp. 367-374). Springer, Cham.
- [16] "Nechynorchuk, O., Kolman, K., Ortolá, M., Persson, M., Holmberg, K., & Bordes, R." (2017). Accelerated ageing of cotton canvas as a model for further consolidation practices. *Journal of Cultural Heritage*, 28, 183-187.

- [17] Hackney S. & Hadley G. (1981). Measurements of the ageing of linen canvas. *Studies in Conservation*, 26(1), 1-14.
- [18] อัมเษฐ์ กรณ์ เขาทอง, & วีร ชัย ชัย วร พฤกษ์. (2016). Study of Air Flow in PVC Canvas Duct.
- [19] Kalair A R, Nileshad S, Ahas N, Sayedmahmoudian M, Stojicevski A. & Koh K. (2021). Application of Business Model Canvas for Solar Thermal Air Conditioners. *Frontiers in Energy Research*, 9.
- [20] Roulos M, N. K. Warren, I. Gong, I. & Yue P. (2010). Web GIS in practice VIII: HTML5 and the canvas element for interactive online mapping.
- [21] "Harrison, C., Tan, D. & Morris, D." (2011). Skinnut: appropriating the skin as an interactive canvas. *Communications of the ACM*, 54(8), 111-118.
- [22] Linterink F. J. & Di Pietro G. (2007). CANVAS PAINTINGS ON COILD WALLS: RELATIVE HUMIDITY DIFFERENCES NEAR THE STRETCHER. *Museum Microclimates. National Museum of Denmark*.
- [23] Stamatoopoulou E, Sotiropoulou M, Karoglou M. & Bakolas A. (2021). Characterization of contemporary artworks made on photosensitized canvas by means of optical microscopy and micro-Raman spectroscopy techniques. *Microchemical Journal*, 165, 106110.
- [24] Stamatoopoulou E, Sotiropoulou M, Karoglou M. & Bakolas A. (2021). Characterization of contemporary artworks made on photosensitized canvas by means of optical microscopy and micro-Raman spectroscopy techniques. *Microchemical Journal*, 165, 106110.
- [25] Akhtar V. & Tvari A. (2015). Study of canvas concrete in civil engineering works. *Int Res J Eng Technol*, 3(1), 314-315.
- [26] Keyser, J. D. & Boetschat, G. (2004). The canvas as the art: landscape analysis of the rock-art panel. *The Figured Landscapes of Rock-art: Looking at pictures in place*, 118-130.
- [27] Hackney S. (1984). The distribution of gaseous air pollution within museums. *Studies in Conservation*, 29(3), 105-116.
- [28] Abd Samad W. A. B. W. & Dahlan A. R. A. An Integrated Risk Management using Business Model Canvas: Designing Business Model Options for The Royal Malaysian Air Force in The Digital Era.
- [29] Hendricky P, Desmarais G, Weder M, Ferreira F. S. & Derome D. (2016). Moisture uptake and permeability of canvas paintings and their components. *Journal of Cultural Heritage*, 19, 445-453.
- [30] Carloni G, Russo M. V, Spinelli E. & Talarico E. (2001). Characterisation of fatty acids in drying oils used in paintings on canvas by GC and GC-MS analysis. *Annali di chimica*, 91(11-12), 719-726.
- [31] Padfield T, Padfield N, Lee D. S. H, Thøgersen A, Nielsen A. V, Andersen C. K. & Scharff M. (2020). Back protection of canvas paintings. *Heritage Science*, 8(1), 1-19.

SCREENSHOTS



```
main.py x
Virtual Paint Ecosystem > main.py > ...
1 import numpy as np
2 import cv2
3 from collections import deque
4
5 def setValues(x):
6     print("")
7
8
9 cv2.namedWindow("Color detectors")
10 cv2.createTrackbar("Upper Hue", "Color detectors", 153, 180, setValues)
11 cv2.createTrackbar("Upper Saturation", "Color detectors", 255, 255, setValues)
12 cv2.createTrackbar("Upper Value", "Color detectors", 255, 255, setValues)
13 cv2.createTrackbar("Lower Hue", "Color detectors", 64, 180, setValues)
14 cv2.createTrackbar("Lower Saturation", "Color detectors", 72, 255, setValues)
15 cv2.createTrackbar("Lower Value", "Color detectors", 49, 255, setValues)
16
17
18 bpoints = [deque(maxlen=1024)]
19 gpoints = [deque(maxlen=1024)]
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
262
```

```

File Edit Selection View Go Run Terminal Help
main.py - Projects - Visual Studio Code

main.py x
Virtual Paint Ecosystem > main.py > ...

34
35
36 paintWindow = np.zeros((471,636,3)) + 255
37 paintWindow = cv2.rectangle(paintWindow, (40,1), (140,65), (0,0,0), 2)
38 paintWindow = cv2.rectangle(paintWindow, (160,1), (255,65), colors[0], -1)
39 paintWindow = cv2.rectangle(paintWindow, (275,1), (370,65), colors[1], -1)
40 paintWindow = cv2.rectangle(paintWindow, (390,1), (485,65), colors[2], -1)
41 paintWindow = cv2.rectangle(paintWindow, (505,1), (600,65), colors[3], -1)
42
43 cv2.putText(paintWindow, "CLEAR", (49, 33), cv2.FONT_HERSHEY_SIMPLEX, 0.5,
44 cv2.putText(paintWindow, "BLUE", (185, 33), cv2.FONT_HERSHEY_SIMPLEX, 0.5,
45 cv2.putText(paintWindow, "GREEN", (298, 33), cv2.FONT_HERSHEY_SIMPLEX, 0.5,
46 cv2.putText(paintWindow, "RED", (420, 33), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (
47 cv2.putText(paintWindow, "YELLOW", (520, 33), cv2.FONT_HERSHEY_SIMPLEX, 0.5
48 cv2.namedWindow('Paint', cv2.WINDOW_AUTOSIZE)
49
50
51
52 cap = cv2.VideoCapture(0)
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

TERMINAL OUTPUT DEBUG CONSOLE
[2/3/2022, 9:58:23 pm] For C source files, IntelliSenseMode was changed from "windows-msv
[2/3/2022, 9:58:23 pm] IntelliSenseMode was changed because it didn't match the detected
[2/3/2022, 9:58:23 pm] For C++ source files, IntelliSenseMode was changed from "windows-m
[2/3/2022, 9:58:23 pm] IntelliSenseMode was changed because it didn't match the detected

C/C++ Configuration W
Ln 11, Col 62 Spaces: 4 UTF-8 CRLF Python

```

```

File Edit Selection View Go Run Terminal Help
main.py - Projects - Visual Studio Code

main.py x
Virtual Paint Ecosystem > main.py > ...

52 cap = cv2.VideoCapture(0)
53
54
55 while True:
56
57     ret, frame = cap.read()
58
59     frame = cv2.flip(frame, 1)
60     hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
61
62
63     u_hue = cv2.getTrackbarPos("Upper Hue", "Color detectors")
64     u_saturation = cv2.getTrackbarPos("Upper Saturation", "Color detectors")
65     u_value = cv2.getTrackbarPos("Upper Value", "Color detectors")
66     l_hue = cv2.getTrackbarPos("Lower Hue", "Color detectors")
67     l_saturation = cv2.getTrackbarPos("Lower Saturation", "Color detectors")
68     l_value = cv2.getTrackbarPos("Lower Value", "Color detectors")
69     Upper_hsv = np.array([u_hue,u_saturation,u_value])
70     Lower_hsv = np.array([l_hue,l_saturation,l_value])
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

TERMINAL OUTPUT DEBUG CONSOLE
[2/3/2022, 9:58:23 pm] For C source files, IntelliSenseMode was changed from "windows-msv
[2/3/2022, 9:58:23 pm] IntelliSenseMode was changed because it didn't match the detected
[2/3/2022, 9:58:23 pm] For C++ source files, IntelliSenseMode was changed from "windows-m
[2/3/2022, 9:58:23 pm] IntelliSenseMode was changed because it didn't match the detected

C/C++ Configuration W
Ln 11, Col 62 Spaces: 4 UTF-8 CRLF Python

```

```

main.py x
Virtual Paint Ecosystem > main.py > ...
69 l_value = cv2.getTrackbarPos("Lower Value", "Color detectors")
70 Upper_hsv = np.array([u_hue,u_saturation,u_value])
71 Lower_hsv = np.array([l_hue,l_saturation,l_value])
72
73
74 frame = cv2.rectangle(frame, (40,1), (140,65), (122,122,122), -1)
75 frame = cv2.rectangle(frame, (160,1), (255,65), colors[0], -1)
76 frame = cv2.rectangle(frame, (275,1), (370,65), colors[1], -1)
77 frame = cv2.rectangle(frame, (390,1), (485,65), colors[2], -1)
78 frame = cv2.rectangle(frame, (505,1), (600,65), colors[3], -1)
79 cv2.putText(frame, "CLEAR ALL", (49, 33), cv2.FONT_HERSHEY_SIMPLEX, 0.5,
80 cv2.putText(frame, "BLUE", (185, 33), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (2
81 cv2.putText(frame, "GREEN", (298, 33), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (
82 cv2.putText(frame, "RED", (420, 33), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (25
83 cv2.putText(frame, "YELLOW", (520, 33), cv2.FONT_HERSHEY_SIMPLEX, 0.5,
84
85
86
87

```

TERMINAL OUTPUT DEBUG CONSOLE

```

[2/3/2022, 9:58:23 pm] For C source files, IntelliSenseMode was changed from "windows-msv
[2/3/2022, 9:58:23 pm] IntelliSenseMode was changed because it didn't match the detected
[2/3/2022, 9:58:23 pm] For C++ source files, IntelliSenseMode was changed from "windows-m
[2/3/2022, 9:58:23 pm] IntelliSenseMode was changed because it didn't match the detected

```

main.py x Python 3.8.5 64-bit

```

main.py x
Virtual Paint Ecosystem > main.py > ...
86
87 Mask = cv2.inRange(hsv, Lower_hsv, Upper_hsv)
88 Mask = cv2.erode(Mask, kernel, iterations=1)
89 Mask = cv2.morphologyEx(Mask, cv2.MORPH_OPEN, kernel)
90 Mask = cv2.dilate(Mask, kernel, iterations=1)
91
92
93 cnts,_ = cv2.findContours(Mask.copy(), cv2.RETR_EXTERNAL,
94 cv2.CHAIN_APPROX_SIMPLE)
95 center = None
96
97
98 if len(cnts) > 0:
99
100     cnt = sorted(cnts, key = cv2.contourArea, reverse = True)[0]
101
102     ((x, y), radius) = cv2.minEnclosingCircle(cnt)
103
104     cv2.circle(frame, (int(x), int(y)), int(radius), (0, 255, 255), 2)

```

TERMINAL OUTPUT DEBUG CONSOLE

```

[2/3/2022, 9:58:23 pm] For C source files, IntelliSenseMode was changed from "windows-msv
[2/3/2022, 9:58:23 pm] IntelliSenseMode was changed because it didn't match the detected
[2/3/2022, 9:58:23 pm] For C++ source files, IntelliSenseMode was changed from "windows-m
[2/3/2022, 9:58:23 pm] IntelliSenseMode was changed because it didn't match the detected

```

main.py x Python 3.8.5 64-bit

```

103
104     cv2.circle(frame, (int(x), int(y)), int(radius), (0, 255, 255), 2)
105
106     M = cv2.moments(cnt)
107     center = (int(M['m10'] / M['m00']), int(M['m01'] / M['m00']))
108
109
110     if center[1] <= 65:
111         if 40 <= center[0] <= 140:
112             bpoints = [deque(maxlen=512)]
113             gpoints = [deque(maxlen=512)]
114             rpoints = [deque(maxlen=512)]
115             ypoints = [deque(maxlen=512)]
116
117             blue_index = 0
118             green_index = 0
119             red_index = 0
120             yellow_index = 0
121
122

```

TERMINAL OUTPUT

```

[2/3/2022, 9:58:23 pm] For C source files, IntelliSenseMode was changed from "windows-msv
[2/3/2022, 9:58:23 pm] IntelliSenseMode was changed because it didn't match the detected
[2/3/2022, 9:58:23 pm] For C++ source files, IntelliSenseMode was changed from "windows-m
[2/3/2022, 9:58:23 pm] IntelliSenseMode was changed because it didn't match the detected

```

main.py - Projects - Visual Studio Code

main.py

EXPLORER

OPEN EDITORS

PROJECTS

- Virtual Paint Ecosystem
- main.py
- weather-app-js
- Zoom Class automate
- desktop.ini
- Untitled1.ipynb

OUTLINE

TIMELINE

Ln 117, Col 31 Spaces: 4 UTF-8 CRLF Python

```

122     paintWindow[67,:,:] = 255
123     elif 160 <= center[0] <= 255:
124         colorIndex = 0 # Blue
125     elif 275 <= center[0] <= 370:
126         colorIndex = 1 # Green
127     elif 390 <= center[0] <= 485:
128         colorIndex = 2 # Red
129     elif 505 <= center[0] <= 600:
130         colorIndex = 3 # Yellow
131     else :
132         if colorIndex == 0:
133             bpoints[blue_index].appendleft(center)
134         elif colorIndex == 1:
135             gpoints[green_index].appendleft(center)
136         elif colorIndex == 2:
137             rpoints[red_index].appendleft(center)
138         elif colorIndex == 3:
139             ypoints[yellow_index].appendleft(center)
140

```

TERMINAL OUTPUT

```

[2/3/2022, 9:58:23 pm] For C source files, IntelliSenseMode was changed from "windows-msv
[2/3/2022, 9:58:23 pm] IntelliSenseMode was changed because it didn't match the detected
[2/3/2022, 9:58:23 pm] For C++ source files, IntelliSenseMode was changed from "windows-m
[2/3/2022, 9:58:23 pm] IntelliSenseMode was changed because it didn't match the detected

```

main.py - Projects - Visual Studio Code

main.py

EXPLORER

OPEN EDITORS

PROJECTS

- Virtual Paint Ecosystem
- main.py
- weather-app-js
- Zoom Class automate
- desktop.ini
- Untitled1.ipynb

OUTLINE

TIMELINE

Ln 117, Col 31 Spaces: 4 UTF-8 CRLF Python

```
File Edit Selection View Go Run Terminal Help
main.py - Projects - Visual Studio Code

main.py x
Virtual Paint Ecosystem > main.py > ...
139 ypoints[yellow_index].appendleft(center)
140
141 else:
142     bpoints.append(deque(maxlen=512))
143     blue_index += 1
144     gpoints.append(deque(maxlen=512))
145     green_index += 1
146     rpoints.append(deque(maxlen=512))
147     red_index += 1
148     ypoints.append(deque(maxlen=512))
149     yellow_index += 1
150
151 points = [bpoints, gpoints, rpoints, ypoints]
152 for i in range(len(points)):
153     for j in range(len(points[i])):
154         for k in range(1, len(points[i][j])):
155             if points[i][j][k - 1] is None or points[i][j][k] is None:
156                 continue
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

TERMINAL OUTPUT DEBUG CONSOLE

C/C++ Configuration W

[2/3/2022, 9:58:23 pm] For C source files, IntelliSenseMode was changed from "windows-msv

[2/3/2022, 9:58:23 pm] IntelliSenseMode was changed because it didn't match the detected

[2/3/2022, 9:58:23 pm] For C++ source files, IntelliSenseMode was changed from "windows-m

[2/3/2022, 9:58:23 pm] IntelliSenseMode was changed because it didn't match the detected

Ln 117, Col 31 Spaces: 4 UTF-8 CRLF Python

```
File Edit Selection View Go Run Terminal Help
main.py - Projects - Visual Studio Code

main.py x
Virtual Paint Ecosystem > main.py > ...
156 if points[i][j][k - 1] is None or points[i][j][k] is None:
157     continue
158 cv2.line(frame, points[i][j][k - 1], points[i][j][k], color
159 cv2.line(paintWindow, points[i][j][k - 1], points[i][j][k],
160
161
162 cv2.imshow("Tracking", frame)
163 cv2.imshow("Paint", paintWindow)
164 cv2.imshow("mask", Mask)
165
166
167 if cv2.waitKey(1) & 0xFF == ord("q"):
168     break
169
170
171 cap.release()
172 cv2.destroyAllWindows ()
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

TERMINAL OUTPUT DEBUG CONSOLE

C/C++ Configuration W

[2/3/2022, 9:58:23 pm] For C source files, IntelliSenseMode was changed from "windows-msv

[2/3/2022, 9:58:23 pm] IntelliSenseMode was changed because it didn't match the detected

[2/3/2022, 9:58:23 pm] For C++ source files, IntelliSenseMode was changed from "windows-m

[2/3/2022, 9:58:23 pm] IntelliSenseMode was changed because it didn't match the detected

Ln 117, Col 31 Spaces: 4 UTF-8 CRLF Python