"Surface Orientation Detection"

Project report submitted in partial fulfillment of the requirement for the degree of Bachelor of Technology

in

Computer Science and Engineering/Information Technology

By

Aayush Singla (151460)
Kanav Gupta (151419)

Under the supervision of

Mr. Kuljeet Singh
(Senior Architect, Wipro Limited)

to

Department of Computer Science & Engineering and Information Technology
Jaypee University of Information Technology Waknaghat, Solan-173234, Himachal Pradesh

Acknowledgements

I express my profound gratitude and indebtedness to of **Mr. Kuljeet Singh,** Senior Architect Wipro limited for introducing the present topic and for their inspiring intellectual guidance, constructive criticism and valuable suggestion throughout this journey.

I am also thankful to all **Mr. Vijay Natarajan** Competency Manager Wipro Limited for his constant motivation and helping me bring in improvements in the project.

Finally I would like to thank my family and friends for their constant support. Without their contribution it would have been impossible to complete my work.

Date

Aayush Singla

Kanav Gupta

Table of Contents

List of the figures

Abstract

This venture was an endeavour at building up an identification and following its direction utilizing current PC vision innovation. The task conveys an executed direction following framework. It comprises of a half breed of optical and present day infrared innovation and is material to territories, for example, unsupervised observation or semi-self-ruling control. It is steady and is relevant as an independent framework or one that could without much of a stretch be inserted into a significantly bigger framework. The undertaking was actualized in 2 months, and included examination into the territory of PC vision and AI. It likewise included the incorporation of bleeding edge innovation of both the equipment and programming kind. The consequences of the task are communicated in this report, and sum to the use of PC vision systems in following vivify protests in both a 2 dimensional and 3 dimensional scene.

Chapter 1: Introduction

1.1 Prelude

This undertaking included actualizing object discovery and its direction following programming. This section gives a talk of the task detail. It likewise gives an abnormal state review of the framework, leaving structure and execution subtleties for discourse in the separate parts. This report tends to the issue of recouping the surface direction of a finished plane from its picture and after that utilizing this data, in a dynamic circumstance, to recuperate the movement parameters of the camera. It comprises of a half and half of optical and present day infrared innovation and is pertinent to territories, for example, unsupervised reconnaissance or semi-self-sufficient control. It likewise gives a guide for the peruse about the general introduction and structure of the report.

1.2 Initialisation & Impression

The task objective was to deliver a working framework for following articles in three-dimensional space. The point of the task was to start from this spec, and plan an answer for the issue. After an attractive arrangement was planned, the assignment came to execute the arrangement. All through the task, numerous issues emerged. These issues fluctuated from execution issues identified with code, and from usage issues identified with impediments of programming advancements utilized. All endeavors at beating these issues are talked about in this report. Likewise, as is normal case with research, as the task advanced, the subsequent following framework displayed in this report contrasts from the underlying plan determination; it is progressively conventional and centers around not following pre-empted questions but rather any items that enter the sensor field of view (fov). Sections 3 and 4 portray the structure and execution of the task and examine the methodology taken to go from detail to the framework displayed nearby this report.

The framework fuses new innovation with sights set on the following issue. The framework is driven by OpenCV Spyder IDE utilizing Anaconda, and various parts of the framework procedure the information in various ways. The focal point of the framework is a client controlled graphical interface for giving input and semi-control of the parts. An abnormal state depiction of the framework is appeared in figure 1.1. The figure portrays the framework as a progression of parts each with their very own particular obligation to

perform. The segments are altogether propelled all the while through the Syder IDE (more detail on this in section 4) and all shutdown when the client flag the GUI to close. The GUI gives the client state-of-the-art following data through the 2D optical tracker and the 3D profundity based tracker. The two trackers give representation capacities, the optical tracker giving a RGB picture stream and the tracker giving a point perception of the scene.

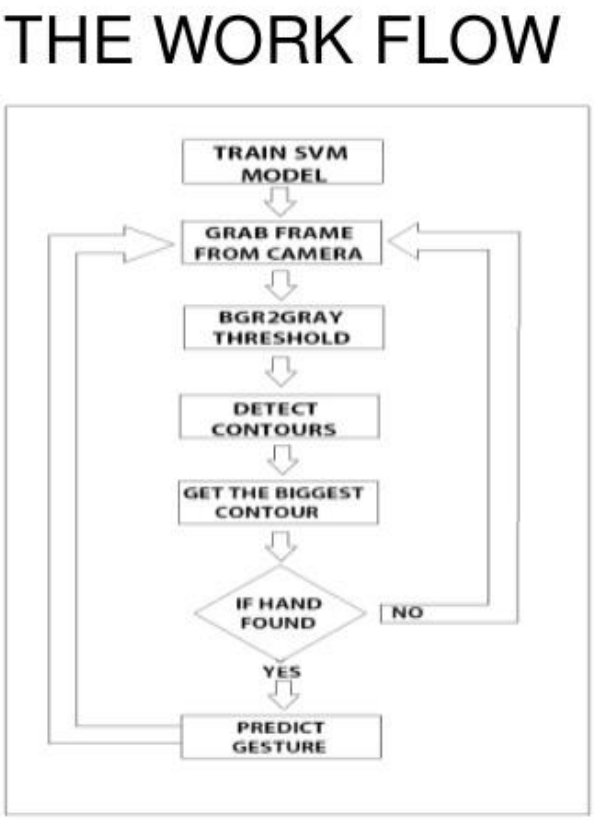

Figure 1.1: The flow diagram of the effective work flow of image processing through OpenCV using Spyder IDE. The GUI is centered, as it provides a means to communicate with the other components.

The following segment gives a prologue to tracking, giving some detail of the product innovation utilized in the venture.

1.3 Tracking

As expressed, the objective of this undertaking was making object location and direction following programming. This undertaking is a progressing point for research, with continous advancement of calculations and execution code. Consequences of this examination have been connected to a scope of items, for example, video reconnaissance, automated vision and independent flight. In mechanical technology, following is regularly used to give a way to confinement and mapping of an obscure domain. One case of this has been the application to empower robots to act human-like and play amusements like soccer [DOL+05]. The following framework executed with this venture fuses the utilization of the propelled PC vision utilizing OpenCV to give ongoing profundity examination.

Utilizing this information, it is conceivable to follow an article in three measurements. Applying a scientific channel to dispose of commotion readings from the sensor, a smooth, solid following framework is actualized. The channel being referred to, a Kalman Filter, is connected in two measurements (xy plane) to smooth aftereffects of the positional information. How this is stretched out to consolidate the 3D information is clarified in section 4.

1.4 Justification

There were various explanations behind endeavour this task. It gave a fascinating subject to investigate, as it consolidated present day sensor innovation with a more established issue. The venture likewise executed an answer that has numerous special traits:

- As the sensor is fairly modest, the venture is a savvy option in contrast to other following frameworks.

- The programming utilizes open-source usage. The sensor is interfaced by means of an open-source library comprising of drivers and an API. At no time is the undertaking upset by restrictive advances.

- The arrangement utilizes the Robot Operating System (ROS, talked about in more detail section 3), an open-source structure for robots. ROS is created by an incredibly

dynamic network and gives top of the line deliberation of many low dimension insights about mechanical autonomy.

It gives an intriguing benchmark to mechanical advancement and PC vision. The task is additionally extendable; An ideal element of the product was to give a control structure to the helicopter alongside the following module. This can be indicated as an augmentation for a future last year venture expanding on the work here (this is talked about further in section 6).

Milieu Studies	Talks about perusing material, innovations utilized and related work. It likewise talks about in detail the calculations that are actualized in the task
Plan & Execution	Plan and structure of the undertaking API's and constraints of programming utilized. Subtleties of utilization code.
Results & Conclusions	Execution benchmarking against a comparable programming usage. Basic audit of the product.
Future Scope	Recommendations for enhancements and future work are proposed here.

All through the task, Object Oriented standards were pursued as most ideal as. In part 3, the undertaking configuration is talked about in detail, utilizing UML charts to demonstrate the practical associations between classes. Documentation was likewise continued, giving a discernible, viable code base. At long last, numerous orders learned all through the UCD software engineering qualification were utilized to actualize the product. This prompted a venture that exhibits a capacity reachable by a software engineering certificate.

1.5 Report Guidance

This record is organized in a top-down design. Part 2 talks about perusing material that demonstrated valuable all through the task. This is continued by center parts which center around the subtleties of the undertaking; structure, usage and results and ends are

altogether exhibited and dissected for the peruser. Table 1.1 gives a sign to the structure of the report.

The finish of the report includes a reference section of any components of the venture that may require additional inclusion, however are outside the extent of the task.

With the appearance of profound learning strategies, the exactness for article identification has expanded radically. The venture means to join best in class procedure for article identification with the objective of accomplishing high precision with a continuous exhibition. A noteworthy test in huge numbers of the item recognition frameworks is the reliance on other PC vision systems for helping the profound learning based methodology, which prompts moderate and non-ideal execution. In this undertaking, we utilize a totally profound learning based way to deal with take care of the issue of article identification in a start to finish design. The system is prepared on the most testing openly accessible dataset (PASCAL VOC), on which an item discovery challenge is led every year. The subsequent framework is quick and precise, in this way supporting those applications which require object location.

Chapter 2: Milieu Studies

2.1 Prelude

Expanding on what has been expressed in the past part, this section subtleties the perusing material that was secured all through the venture. As hindrances were met in the execution of the venture, it was important to re-evaluate specific regions of code (calculations, nearby improvements and so on.) and change the usage to take into consideration better execution among different expansions. Research and perusing did as the foundation explore identified with the venture are portrayed. This section will diagram key papers that depicted the general PC vision methods incorporated into the undertaking and thinks about what was contemplated.

2.2 Technology

Above all else to comprehend my task you need to comprehend PC vision. Item following and discovery includes in succession is a significant and essential issue in PC vision. This

territory of research has a great deal of utilizations in face ID frameworks, model based coding, look discovery, human PC, association, video chatting, and so on.

2.2.1 OpenCV

OpenCV implies Intel O S(open-source) C.V. (Computer Vision) Library. It is a gathering of C capacities and a couple C++ classes that execute some well known Image Processing and C.V. calculations. OpenCV has cross-stage center to-abnormal state API that comprises of a couple of hundreds C capacities. It doesn't depend on outside libraries, however it can utilize some when it is conceivable. OpenCV is free for both non-business and business use. OpenCV gives straightforward interface to Intel Integrated Performance Primitives (IPP). That is, it stacks consequently IPP libraries enhanced for explicit processor at runtime, on the off chance that they are accessible.

Alongside entrenched organizations like Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda, Toyota that utilize the library, there are numerous new businesses, for example, Applied Minds, Video Surf, and Zeitera, that utilize OpenCV. OpenCV's conveyed uses length the range from sewing street view pictures together, identifying interruptions in observation video in Israel, checking mine hardware in China, helping robots explore and get objects at Willow Garage, discovery of pool suffocating mishaps in Europe, running intelligent workmanship in Spain and New York, checking runways for flotsam and jetsam in Turkey, examining names on items in production lines the world over on to quick face identification in Japan.

2.2.2 What’s new in OpenCV

- More camera supports the high gui. Various variants of libdc would now be able to be utilized.

- More kinds of video documents would now be able to be perused by utilizing libavcodec from the ffmpeg-0.4.

- Python wrappers in Open CV was made by O. Bornet and M. Asbach using SWIG.
The wrappers ought to be OS-free, They are based on Linux.

- Open CV manufactures and keeps running on 64 bit stages. Additional setups was added to extend records in MsDevStudio 6.0.

- Performance tests and part of cv has been made. the yield position is plain csv and is like the one utilized in IPP. run cxcoretest - t and cvtest - t.

- Training now consequently delivers .xml database alongside the typical index tree.

- Script for making custom powerful library for a subset of IPP, utilized by OpenCV, has been made. Take a gander at opencv/interfaces/ipp

- Camera alignment and epipolar geometry capacities have been totally modified, API was streamlined, and docs refreshed.

- New checkerboard recognition calculation (in light of Vladimir Vezhnevets' code) is currently utilized.

- cvCvtColor underpins new shading models (HLS, CIE L*u*v*), for each RGB->something change the reverse is given, 32f arrangement is totally upheld, 16u is somewhat bolstered.

- Distance change was reached out to discover the closest associated segment of zero pixels for each pixel, not just the separation to it.
- Highgui now recollects places for opened windows in library.

2.2.3 Image Processing

PC control of pictures. A portion of the numerous calculations utilized in picture preparing incorporate convolution, FFT, DCT, diminishing ,edge identification and difference upgrade. These are normally actualized in programming yet may likewise utilize unique reason equipment for speed. Picture handling appears differently in relation to PC illustrations, which is typically increasingly worried about the age of counterfeit pictures, and perception, which endeavors to see (genuine world) information by showing it as a fake picture (for example a diagram). Picture preparing is utilized in picture acknowledgment and PC vision. Silicon Graphics fabricate workstations which are regularly utilized for picture preparing. There are a couple of programming dialects intended for picture handling, for example CELIP, VPL, C++, Python.

2.3 Algorithms and Exploration

Just as understanding the product and equipment innovation utilized in PC vision, broad foundation perusing was done on the calculations and strategies generally utilized in PC vision applications. This area gives a review of a portion of the papers read and gives references of the papers noted. For certain areas that require some additional clarifying, the peruser is eluded to the proper informative supplement segment.

2.3.1 Object recognition in 2D: Foreground & Background Segmentation

Numerous calculations are there for recognizing objects in an information stream. One such calculation that was inquired about and tried for this undertaking. Recordings containing foundation protests under different conditions relating to lighting and development were prepared and moving articles were fragmented from the closer view dependent on arrangement under Bayesian principles. The foundation is displayed by the likelihood of a pixel being out of sight or closer view. This is utilized in deciding component vectors and arranging by the an earlier likelihood.

Highlight vectors picked for every pixel essentially comprise of shading arrangement. Shading co-occurance , where a gathering of neighbouring pixels can have a similar shading properties, is likewise joined by keeping up the qualities at $t = t - 1$. This is organized into the component insights table appeared by HLGHT03 and parameters for

the quantization of the picture pixels and the span of the measurements table which is number of highlights to consider were picked through examination to yield great outcomes.

The fundamental calculation in the paper depends on the factual displaying given, where various highlights are distinguished by following the means of the calculation. The calculation can be separated into four stages (Detailed depiction is precluded here as it isn't applicable; the peruser may wish to allude to the reference):

•Pixels of unimportant change are sifted through fleeting differencing ,where examination of past state can be utilized to distinguish pixels that haven't changed. Pixels are ordered by this foundation and transient differencing.

•If a pixel has been identified as being transient, it is delegated a movement pixel and is additionally arranged by highlight vector correlation and reference to the element table as being closer view or foundation.

•The point set are smoothed which is applied a channel activity to any wrongly marked foundation pixel to portion the frontal area objects.

•The foundation reference picture is refreshed each time another purpose of 'unimportant change' is found.

The paper at that point proceeds to think about the outcomes acquired against benchmarked calculations primarily identifying with Gaussian conveyance strategies. This algorithm was initially planned for use in the venture however was exchanged after some testing for the calculation underneath. The accompanying calculation gave a less definitive tracker, however a snappier edge rate.

2.3.2 Object recognition in 2D: Adaptive Background Mixture Model for Real-Time Tracking

The calculation picked to actualize the 2D following framework was a usage of the calculation proposed by KB01. This calculation is an enhancement for SG99, where the creators executed an ongoing quality set so as to stay up with the latest. The first calculation utilized a without any parameters updating using thresholding by keeping up a foundation model of the scene through a reference picture. Foundation subtraction is handled on the scene to section the frontal area objects. The foundation model depends on a blend of Gaussian circulations, and various loads are utilized to speak to the transient

segments of hues in the scene. When denoting a pixel, loads are connected to each Gaussian, and the one with the most plausible result is utilized to check the pixel. As Bowden et al. guarantee, this technique was unfavourably influenced by the stochastic update capacities used to refresh the foundation model. Bowden et al. enhanced this by utilizing a desire expansion approach that utilizes update capacities considering the L most recent windows of Gaussian models.

This calculation includes in the 2D object following element of the venture by means of an OpenCV execution of a mass tracker. This mass tracker was utilized to gauge the area of the helicopter in 2D and forward handled yield to the 3D tracker.

2.3.3 Kalman Filter

Another significant part of the venture was taking care of commotion yield from the sensor. All information yield from the Kinect is liable to commotion, and thus it was important to process the information through a channel.

A very referred to channel is the Kalman channel. It is a way to figure, with negligible blunder, the state where a procedure dwells. It utilizes numerical capacities to gauge the condition of an item (or all the more uniquely a procedure) before, present and future and it can do this when given an obscure model. It is a molecule separating method, a methods for assessing the genuine way of a model framework (for this situation an object) amid its development. Working on a discrete-time controlled procedure [WB06], it will consider the separating of loud information gotten by the following arrangement of the helicopter. The contributions to the channel are the (loud) current x-y-z co-ordinates of the helicopter, the direction and maybe the speed/quickening, while the yield is an expected state. This state is illustrative of the followed helicopter, and can be utilized to register the future condition of the framework.

The channel has various parts. It monitors its past state x0 in time step k − 1, alluded to as the from the earlier state [WB06], and, utilizing a given estimation z, it attempts to evaluate its present state x00 at step k. With these qualities come edges for blunder, and these are spoken to as straight conditions. These blunders are known as the from the earlier and a posterior gauge mistakes [WB06]. The blunder covariance would then be able to be inferred as the grid comprising of the mistake gauge and its transpose. At long last every one of these qualities are utilized in the accompanying calculation which executes the Kalman channel (appeared as pseudo-code):

Additionally, the channel has a condition for speaking to the distinction in separation between the genuine esteem and the anticipated esteem called the leftover [WB06]. In the event that this esteem is to be zero whenever, this implies the two qualities are in understanding, and the odds of accomplishing an exact sifting is high. The condition for figuring the new state includes the past state and straight mix of the subsequent distinction between perceptions, weighted to a factor K [WB06]. Welch et al characteristic the weighted factor K to depicting the dependableness of the gauge. As the mistake covariance for the estimation approaches zero, the real estimation got as contribution to the separating step can be trusted with being all around likely. On the other hand, if the from the earlier blunder covariance esteem falls, the genuine gauge can be viewed as less exact and the anticipated esteem ought to be considered as progressively solid [WB06].

In its second a large portion of, the paper on the prologue to the Kalman channel starts to depict the all-encompassing Kalman Filter. The all-encompassing rendition of the channel is utilized when the estimation procedure relationship isn't direct [WB06]. This task concentrated on following moving articles space. The estimations taken to express their state were estimations of position and direction, both straight balances. The Kalman filter includes in the optical tracker segment of the undertaking.

2.3.4 Canny-Edge Detection

The Canny edge identifier is an edge acknowledgment overseer that uses a multi-compose figuring to recognize a wide extent of edges in pictures. It is a strategy to isolate significant essential information from different vision objects and definitely decrease the proportion of data to be dealt with. It has been commonly associated in various PC vision structures. Careful has found that the necessities for the use of edge area on different vision systems are reasonably practically identical. Accordingly, an edge acknowledgment answers for area these necessities can be realized in a wide extent of conditions. The general criteria for edge area include:

1. Detection of edge with low mistake rate, which implies that the location ought to precisely get however many edges appeared in the picture as could reasonably be expected

2. The edge point identified from the administrator ought to precisely limit on the focal point of the edge.

3. A given edge in the picture should just be stamped once, and where conceivable, picture clamour ought not make false edges.

Among the edge discovery techniques grew up until this point, Canny edge recognition calculation is a standout amongst the most carefully characterized strategies that gives great and solid identification. Inferable from its optimality to meet with the three criteria for edge identification and the effortlessness of procedure for usage, it wound up a standout amongst the most well known calculations for edge discovery.

The Process of Canny edge location calculation can be separated to 5 distinct advances:

1. Apply Gaussian channel to smooth the picture so as to expel the clamour
2. Find the power angles of the picture
3. Apply non-most extreme concealment to dispose of deceptive reaction to edge identification
4. Apply twofold limit to decide potential edges
5. Track edge by hysteresis: Finalize the recognition of edges by stifling the various edges that are frail and not associated with solid edges.

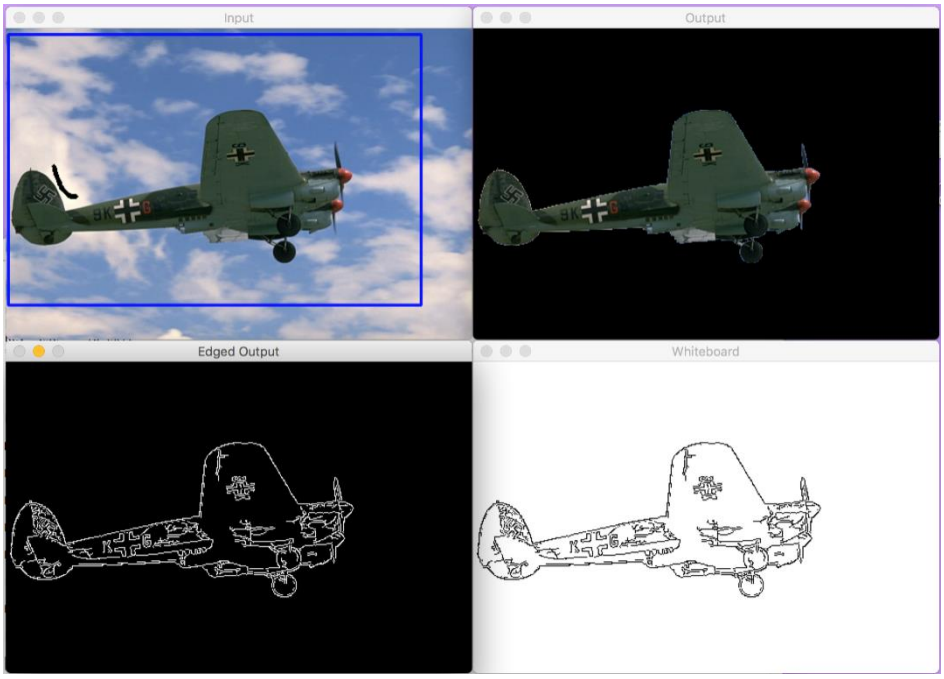

Figure 2.1: Shows the implementation of Canny-Edge Detection on an Airplane where we have to select the foreground and background with precision.

2.3.4.1 Improvement on Canny Edge Detection

While conventional Canny edge discovery gives generally straightforward however exact technique for edge location issue, with all the more requesting prerequisites on the exactness and vigour on the identification, the customary calculation can never again handle the difficult edge recognition task. The fundamental imperfections of the conventional calculation can be condensed as pursues:

1. A Gaussian channel is connected to smooth out the clamour, however it will likewise smooth the edge, which is considered as the high recurrence include. This will build the likelihood of missing frail edges, and the presence of disengaged edges in the outcome.

2. For the angle plentifulness figuring, the old Canny edge location calculation utilizes the inside in a little 2×2 neighbourhood window to ascertain the limited contrast mean an incentive to speak to the inclination adequacy. This technique is delicate to commotion and can undoubtedly distinguish false edges and lose genuine edges.

3. In the conventional Canny edge identification calculation, there will be two fixed worldwide limit esteems to sift through the bogus edges. In any case, as the picture gets mind boggling, distinctive neighbourhoods need altogether different edge esteems to precisely locate the genuine edges. Likewise, the worldwide limit esteems are resolved physically through analyses in the conventional strategy, which prompts unpredictability of computation when countless pictures should be managed.

4. The consequence of the customary location can't achieve an attractive high precision of single reaction for each edge - multi-point reactions will show up.

2.3.5 GrabCut Algorithm

GrabCut is an image division procedure subject to chart cuts. Starting with a customer decided bobbing box around the article to be divided, the computation surveys the shading dispersal of the target thing and that of the establishment using a Gaussian mix model. This is used to build up a Markov discretionary field over the pixel names, with essentialness work that lean towards related regions having a comparative name, and running a diagram cut based improvement to translate their characteristics. As this

measure is presumably going to be more exact than the principal, taken from the hopping box, this two-advance strategy is repeated until association.

Evaluations can be additionally adjusted by the client by pointing out misclassified areas and rerunning the improvement. The strategy likewise rectifies the outcomes to save edges.

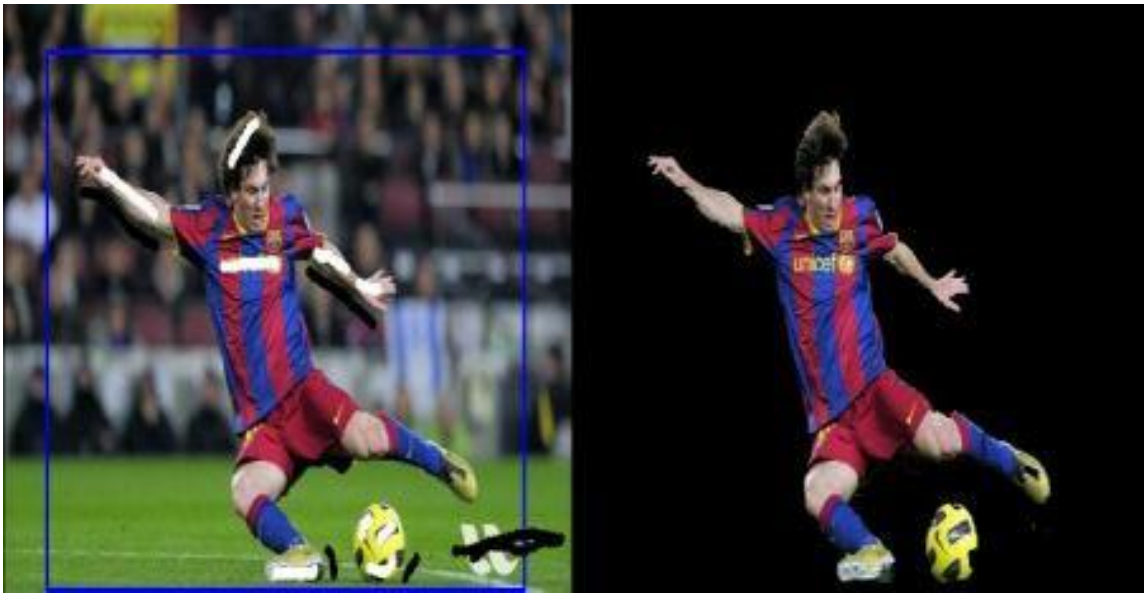

Figure 2.2: Shows the implementation of Image Segmentation using GrabCut Algorithm using OpenCV.

2.3.6 Shi- Tomasi Algorithm

The Shi-Tomasi corner identifier depends altogether on the Harris corner locator. In any case, one slight variety in a determination criteria improved this identifier much than the first. It works very well where even the Harris corner locator comes up short. So here's the minor change that Shi and Tomasi did to the first Harris corner finder.

2.3.6.1 The Alteration

The Harris corner finder has a corner determination criteria. A score is determined for every pixel, and if the score is over a specific esteem, the pixel is set apart as a corner. The score is determined utilizing two eigenvalues. That is, you gave the two eigenvalues to a capacity. The capacity controls them, and gave back a score. Shi and Tomasi recommended that the capacity ought to be discarded. Just the eigenvalues ought to be

utilized to check if the pixel was a corner or not. The impact locale differs as beneath-

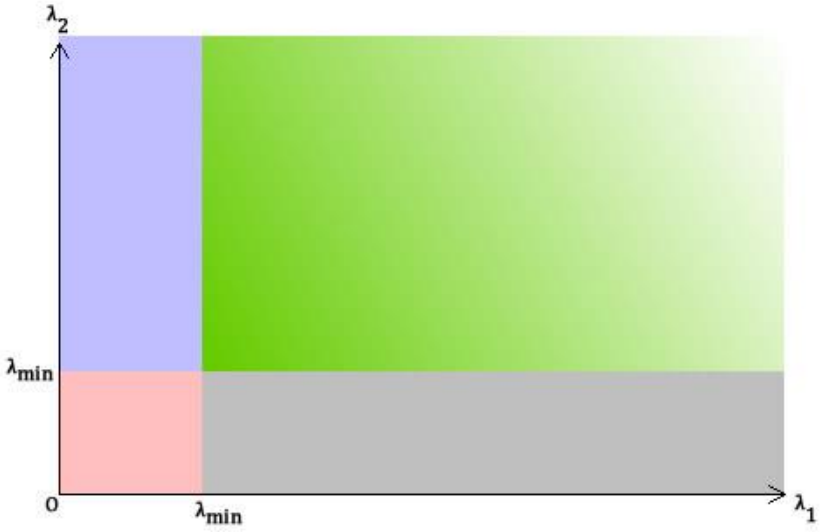

Figure 2.3: In this figure, when λ1 and λ2 are above a threshold value, λmin is a corner which is in green region.

- Green: both λ_1 and λ_2 are greater than a threshold value. Thus, this region is for pixels accepted as corners.
- In the blue and gray regions, either λ_1 or λ_2 is less than he required minimum.
- In the red region, both λ_1 and λ_2 are less than the required minimum. Compare the above with a similar graph for Harris corner detector. You will see the blue and grey areas are equivalent to the edge areas. The red region is for flat areas. The green is for corners.

OpenCV has a function, cv2.goodFeaturesToTrack(). It discovers N most grounded corners in the picture by Shi-Tomasi strategy (or Harris Corner Detection, in the event that you indicate it). Not surprisingly, picture ought to be a grayscale picture. At that point you determine number of corners you need to discover. At that point you determine the quality dimension, which is an incentive between 0-1, which means the base nature of corner beneath which everybody is rejected. At that point we give the base Euclidean separation between corners recognized.

With every one of these informations, the capacity discovers corners in the picture. All corners beneath quality dimension are rejected. At that point it sorts the rest of the corners dependent on quality in the diving request. At that point work takes first most grounded

corner, discards all the adjacent corners in the scope of least separation and returns N most grounded corners. Execution of Shi-Tomasi Algorithm is as per the following-

Spyder (Python 3.7)

File Edit Search Source Run Debug Consoles Projects Tools View Help

Editor - C:\Users\kanav\untitled0.py

temp.py untitled0.py

```
1 import numpy as np
2 import cv2
3 from matplotlib import pyplot as plt
4
5 img = cv2.imread('Output.jpeg')
6 gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
7
8 corners = cv2.goodFeaturesToTrack(gray,25,0.2,10)
9 corners = np.int0(corners)
10
11 for i in corners:
12     x,y = i.ravel()
13     cv2.circle(img,(x,y),3,255,-1)
14
15 plt.imshow(img),plt.show()
```

Figure 2.4: Code implemented for Shi-Tomasi Algorithm

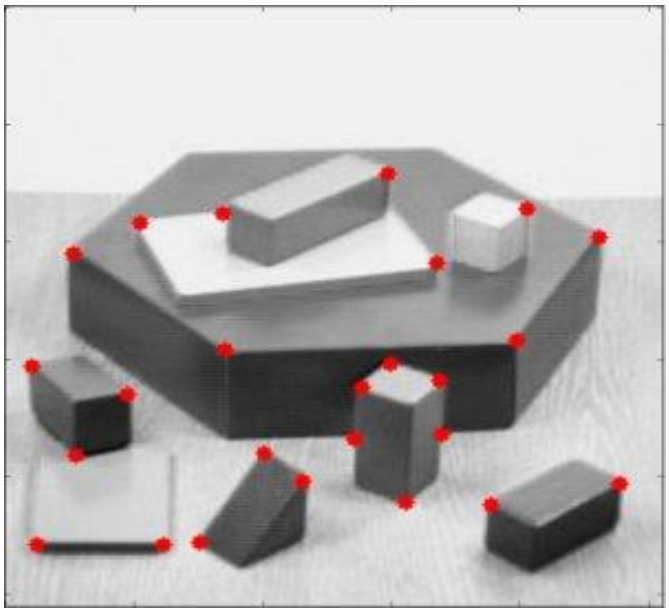

Figure 2.5: Shows the resulted output image after implementing Shi-Tomasi Algorithm and we can clearly see the corners detected using this methodology.

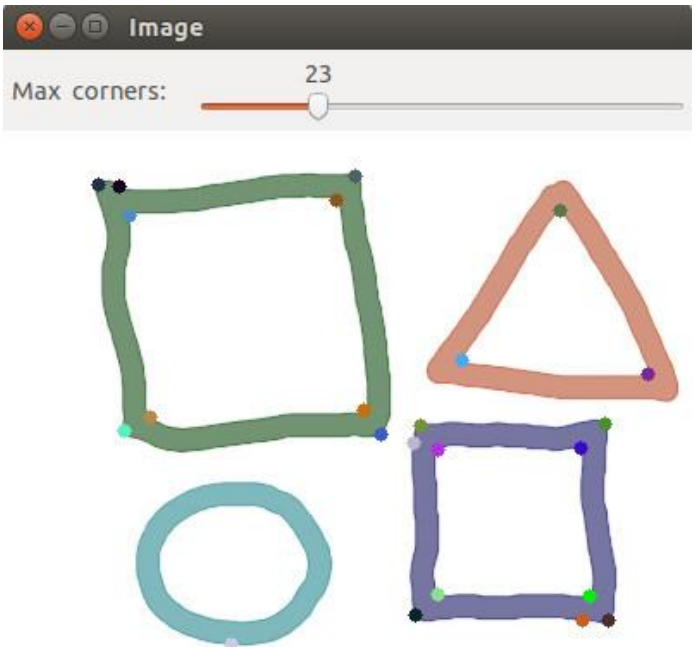

Figure 2.6: Another example of Shi- Tomasi Corner Detection

Chapter 3: Design

3.1 Prelude

This part presents the undertaking formally, giving a determination of the plan and talks about the thoughts and execution issues that must be considered with this structure. It traces purposes behind the incorporation of specific innovations and their suitability to the main job. The section likewise talks about structure issues from a product point of view; programming systems that were clung to however much as could be expected, for example, Object Oriented Design standards, Test Driven Development, programming documentation and rendition control programming. At last, it closes with certain remarks about the plan and how the venture result was influenced by structure choices made right off the bat.

3.2 Design Specification

As expressed already, the task particular was to research and assemble a framework for powerful following of an article in 3 dimensional space. All the more formally, the goal was for a product arrangement which would empower ongoing following of this article which would then eventually prompt the establishments for a completely robotized control framework. So as to accomplish this objective, the undertaking configuration needed to catch all parts of the framework at run-time, and speak to the progression of information through the framework. Figure 1.1 in part 1 demonstrates an abnormal state layout of the framework.

The framework was initially planned, from a product viewpoint, as a solitary procedure application that kept running in a multi-strung condition. This implied all parts of the framework kept running in isolated strings, with the fundamental string being dispensed to the GUI. While this structure gave a straightforward way to actualize the framework, there were numerous issues with it.

Considering this, clearly an update was all together. This procedure buys in to every distributed datum, giving the client' of the framework with ongoing criticism of the trackers and enabling them to shutdown the framework.

Chapter 4: Project Implementation

4.1 Prelude

The objective of this chapter is to get familiar with how we have implemented the code in this project and how we have used the tools required for the project. We have implemented our project using python language.

4.2 Procedures

Following are the procedures that will be implemented for the completion of the project in chronological order

- We have gone through the python, OpenCV.
- In OpenCV the topics which we need to go through were Image Segmentation, Thresholding, Grab cut algorithm, Canny Edge detection algorithm, Shi Tomasi algorithm, Contours in OpenCV.
- High resolution images of the airplane standing in different directions are taken from various image sources like google, Pinterest.
- The tailfin is segmented out using various image processing tasks like Thresholding, smoothening, Grab cut algorithm, Canny Edge detection algorithm, Contours in OpenCV.
- Then we have segmented out the corners of the tailfin using Shi-Tomasi algorithm.
- After segmenting out the corners we joined the lines with those points.
- Finding the two upper angles (one will be obtuse and the other will be acute) and the length of the edges of the tailfin.
- By default, the ratio of the obtuse and acute angle is known of the respective airplane's tailfin. If at a particular position the ratio of the angles matches with the default value then the airplane is standing in the horizontal position and to find the direction, we need to compare the lengths of the edges. If the left edge length is

greater than the right edge then the airplane is standing in the east and if the left edge length is lesser than the right edge then the airplane is standing in the west.

- Calculating the ratio between those two angles i.e. obtuse/acute.
- If the ratio is greater than the default angle ratio and the left edge is greater than the right edge, the airplane is standing in the south-east direction. Similarly, if the ratio is greater than the default angle ratio and the left edge is smaller than the right edge, the airplane is standing in south-west direction.
- If the ratio is smaller than the default angle ratio and the left edge is greater than the right edge, the airplane is standing in the north-east direction. Similarly, if the ratio is smaller than the default angle ratio and the left edge is smaller than the right edge, the airplane is standing in north-west direction.

4.3 Python

Python programming was created by Guido Van Rossum and came in 1991.
Python is a

- General purpose
- Dynamic
- High level
- Interpreted programming language
- Supports Object Oriented programming (OOP) approach
- Simple and also very easy to learn
- Powerful and versatile

Some of the Python applications are-:

- It is used on a web server to create different apps.
- It is used by the software to create work flows.
- It connects with the database.
- It is used in big data to handle huge amount of data.
- It is used as a scripting language.

OpenCV supports many programming languages such as C++, Python, Java, etc. OpenCV-Python is a library in Python which is designed to solve computer vision

problems. It is the Python API for OpenCV which combines OpenCV C++ API and the Python language. So we have used OpenCV library in python.

4.3.1 OpenCV

Open Source Computer Vision Library (OpenCV) is a computer vision which is an open source and easily available and also has machine learning software library in it. It was first started at Intel in 1999 by Gary Bradsky but the release came in 2000. OpenCV supports many algorithms related to Computer Vision and Machine Learning and is increasing day by day.

OpenCV is used to provide a common infrastructure for computer vision applications. It is a BSD-licensed product which means it makes it easy for businesses to use the code. OpenCV includes several hundreds of computer vision algorithms.

OpenCV is a modular structure that is it has packages which includes several shared or static libraries. The different modules are available such as:

- **Core functionality** – It is a module which has basic data structures, and also multi-dimensional array which can be used by different modules.
- **Image Processing** - It includes all the image related things some ofv those are linear/ non-linear image filtering, different image transformations, color space conversion, histograms, and many other.
- **Video Analysis** - It includes many features like estimating motion of objects, background removal, and tracking algorithms for a particular objects.
- **Camera Calibration and 3D Reconstruction** – It has different view geometry algorithms, calibration with single and stereo cameras, estimating different poses of an object, and elements of 3D reconstruction.
- **2D Features Framework** –It has feature detectors as well as descriptors.
- **Object Detection** – It is used for detection of objects and also things like eyes, faces, cars, people, planes and many other things).
- **High-level GUI** - an easy-to-use interface to simple UI capabilities.
- Some other useful things that can be useful are Google test wrappers, Machine Learning, FLANN, Computational Photography Python bindings, and others.

4.3.2 Image Segmentation

In computer vision, image segmentation means that on some particular criteria we are dividing the image into different group of pixels. In an image segmentation algorithm a particular image is taken as an input and it gives an output as a collection of different regions or different segments which can be represented as

A collection of different contours or a mask that is either grayscale or color where each and every segment is given a unique grayscale value or color to identify it.

There are different problems of image segmentation such as it has a graph partitioning problem, it has an energy minimization problem but it is used as a solution to a various partial differential equations.

Applications of Image Processing

The information which are visualized are the most important information which can be used for the interpretation and processing them. In today's world we want everything should be automated and the information provided can be used in things like television, robotics, photography etc.

- Photoshop which are computerized photography.
- In space we can use image processing which is known as space image processing.
- Image processing can be used in medicinal things like interpretation of X-ray images.
- Automatic characters can be identified such as zip codes.
- Finger prints or face can be recognized.

In our project we used OpenCV an inbuilt library in python for the image processing. There are different functions and things that are used in our project which are listed below.

- **Reading an Image**

  First we have to import the library of OpenCV in python and then we will read the image using function cv.imread().

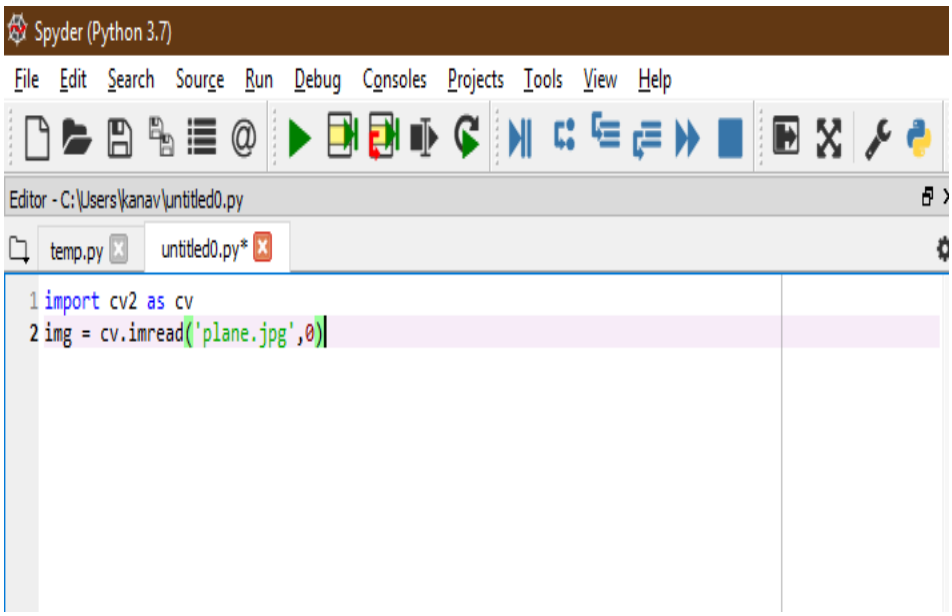

Figure 4.1: This figure Shows how to read an image.

- **Display an Image**

  To show an image functions used are cv.imshow()-shows the image, cv.waitkey(0)-shows image until you press any key on keyboard, cv.destroyAllWindows()-closes all the windows.

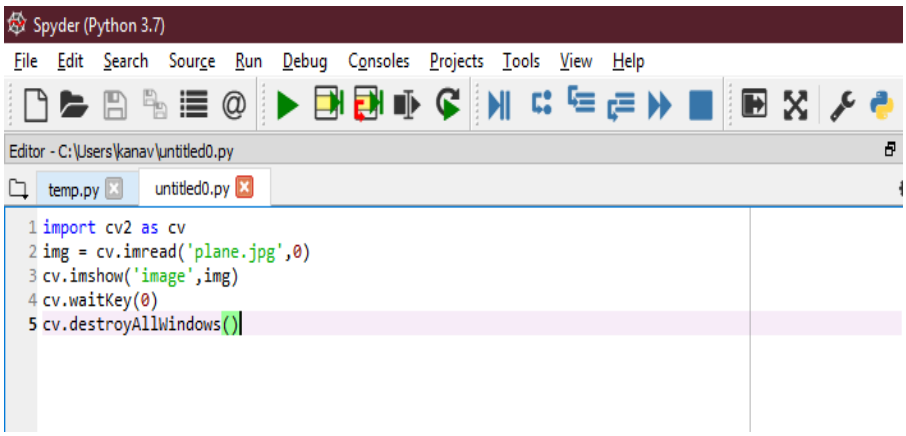

Figure 4.2: This figure Shows how to show an image.

- **Write an Image**

  To write an image function used is cv.imwrite().

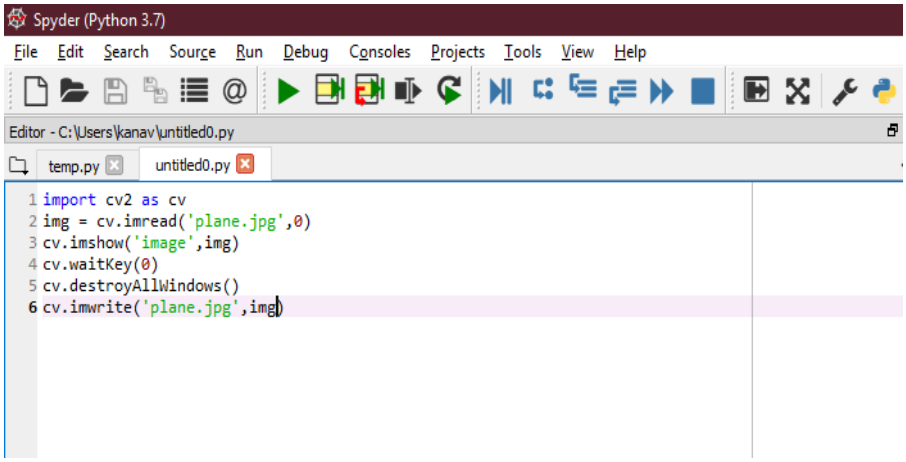

Figure 4.3: This figure Shows how to save an image.

- **Drawing Rectangle**

  To draw a Rectangle we use function cv.rectangle().

- **Mouse Events**

  There are different mouse events which are used in OpenCV such as

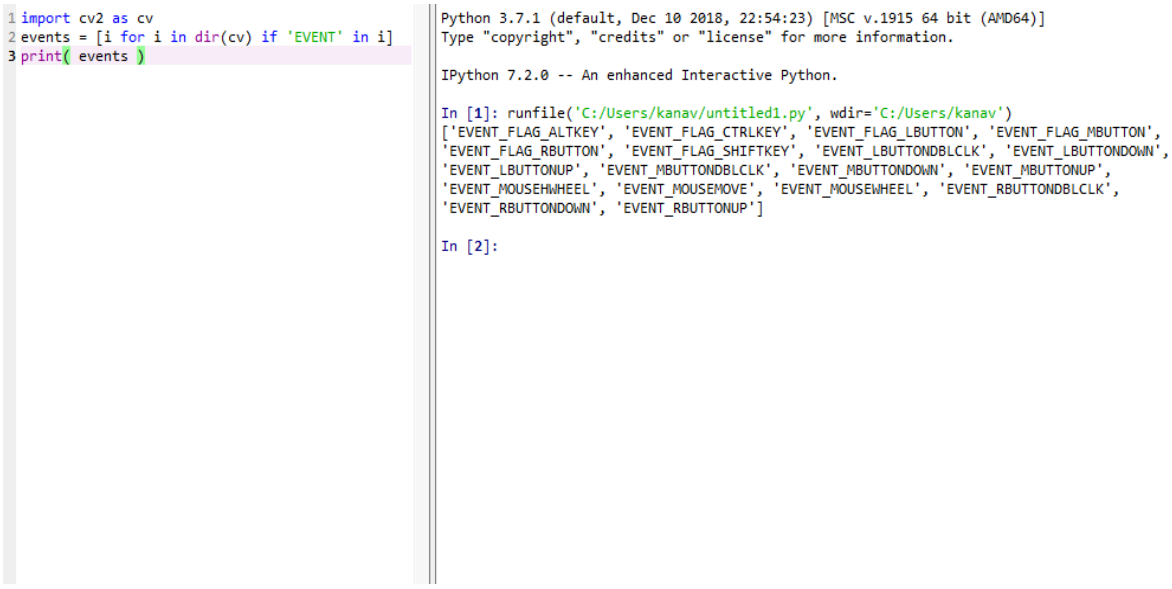

Figure 4.4: This figure Shows the code which gives all the mouse events.

From many of the mouse events we used some of them in our project.

- **Smoothing Images**

  To smoothen an image we will use a function cv.filter2D(). This function will average out the image.

- **Changing color spaces**

  There are 150+ color-space conversion methods in OpenCV. But widely used are, BGR to HSV and BGR to Gray .

  The function used for this is cv.cvtColor(input image, Type Conversion)

  For BGR to Gray conversion we use cv.COLOR_BGR2GRAY.

  For BGR to HSV, we use the flag cv.COLOR_BGR2HSV.

Object Tracking

We will convert BGR image to HSV because this is important for extracting a colored object. Following is the method in which we can do this conversion:

- Take all frames from the video/picture.
- Convert BGR to HSV color-space using the above method function
- Then we give some threshold value to the image for a range of some color
- Now extract what we want to extract.

4.4 Thresholding

- Simple Thresholding

  For every pixel, there must be some threshold value. If the value of the pixel is smaller than the threshold, its value is 0, otherwise it is 1. The function which is used is cv.threshold(, , ,). The first parameter is the source of an image, which is a grayscale image. The second Parameter is the threshold value which is given by the user according to his needs. The third parameter is the maximum value but in our case it is 1 by default and not given by the user. OpenCV has different types of thresholding and which is given by the last parameter .Types of simple thresholding are:

  cv.THRESH_BINARY

cv.THRESH_BINARY_INV

cv.THRESH_TRUNC

cv.THRESH_TOZERO

cv.THRESH_TOZERO_INV

It returns two outputs. The first is the value of the threshold used and the second is output image.

- Adaptive Thresholding

  This gives better results than simple thresholding It has one parameter less as it does not take globally the value from the user it only takes maximum value. cv.adaptiveThresholding(, ,) function is used. It has two types

  cv.ADAPTIVE_THRESH_MEAN_C: The threshold value = mean of the neighborhood area - the constant C.

  cv.ADAPTIVE_THRESH_GAUSSIAN_C: The threshold value = gaussian-weighted sum of the neighborhood values - the constant C.

- Otsu’s Binarization

  In worldwide thresholding, we utilized a self-assertive picked an incentive as a limit. Interestingly, Otsu's strategy abstains from picking an esteem and decides it consequently.

  Consider a picture with just two unmistakable picture esteems (bimodal picture), where the histogram would just comprise of two pinnacles. A decent limit would be amidst those two qualities. So also, Otsu's technique decides an ideal worldwide edge an incentive from the picture histogram.

  So as to do as such, the cv.threshold() work is utilized, where cv.THRESH_OTSU is passed as an additional banner. The limit esteem can be picked subjective. The calculation at that point finds the ideal limit esteem which is returned as the principal yield.

This area exhibits a Python execution of Otsu's binarization to indicate how it functions really. In the event that you are not intrigued, you can avoid this.

Since we are working with bimodal pictures, Otsu's calculation endeavors to discover a limit esteem (t) which limits the weighted inside class change given by the connection:

$$\sigma 2w(t)=q1(t)\sigma 21(t)+q2(t)\sigma 22(t)$$

where

$$q1(t)=\sum i=1tP(i)\&q2(t)=\sum i=t+1IP(i)$$

$$\mu 1(t)=\sum i=1tiP(i)q1(t)\&\mu 2(t)=\sum i=t+1IiP(i)q2(t)$$

$$\sigma 21(t)=\sum i=1t[i-\mu 1(t)]2P(i)q1(t)\&\sigma 22(t)=\sum i=t+1I[i-\mu 2(t)]2P(i)q2(t)$$

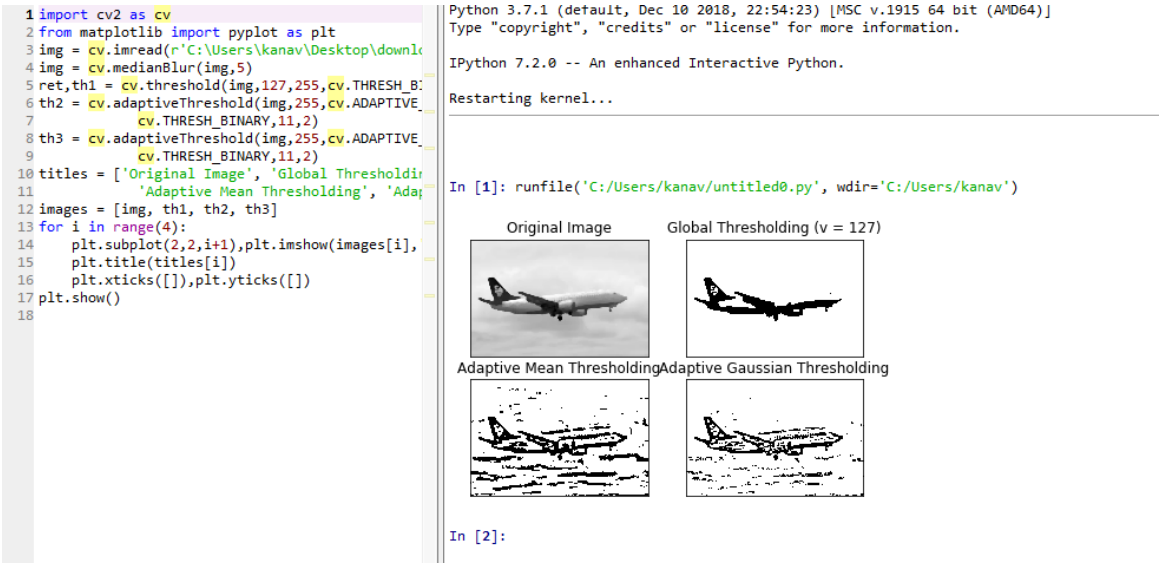

Figure 4.4: Different types of thresholding applied to our image of airplane.

4.5 Canny-Edge Detection

It is an edge detection algorithm. It was developed by John F. Canny. There are different stages of this algorithm.

- Noise Reduction

  We have to reduce the noise in the image with 5*5 Gaussian filter. cv.GaussianBlur() is the function which is used for this.

- Finding Intensity Gradient of the Image

  Then we get the first derivative in the horizontal direction which is represented as Gx and in vertical direction (Gy).

  Edge Gradient(G) = $(Gx^2 - Gy^2)^{\frac{1}{2}}$

  Angle(θ)=$\left(\tan\left(\frac{\mathrm{Gy}}{\mathrm{Gx}}\right)\right)^{-1}$

- Non-maximum Suppression

  Then we will do a full scan of image in order to remove the unwanted pixels. For this, all the pixels are checked whether or not it is a local maximum. The final result that we will get is a binary image with very thin edges.

- Hysteresis Thresholding

  This is an important stage it decides whether the edges are there or not. We give two threshold values, minimum Value and maximum Value. The edges which are more than maximum Value are edges and those below minimum Value are non-edges, so they are left. Those which are in between these two thresholds are considered according to their connectivity. If they are related to edge pixels, they are part of edges else they are also left out. So get is strong edges in the image.

OpenCV combines all these steps in a single cv function which is cv.Canny(). This function has 3 parameters minimum value, maximum value and aperture size whose default value is 3.

4.6 Contours

These are the curve that joins all the continuous points including the boundaries also which have same color or intensity. These are useful tools to detect the shape or anything in the object.

We are using binary images and to find the contours we applied canny edge detection. In OpenCV, to find contour we differentiate between black and white backgrounds. So to find tailfin, it should be in white and background should be in black.

The function used is cv.findContours(), there are three parameters- source image, contour retrieval mode, **contour approximation method**. And output is the hierarchy and contours. Contours are a Python list of the image. Every single contour is a Numpy in built library in which there is array of (x,y) coordinates.

cv.drawContours() function is used to draw contours. It has two parameters- source image, contours are passed as a Python list, index of contours, if we need all contours its value is -1 and other arguments such as thickness, color etc.

Draw all the contours of an image: cv.drawContours(, , -1, ,)

Draw individual contour, suppose 4th contour: cv.drawContours(, , 3, ,)

Most useful method: cnt = contours[4]

cv.drawContours(, [cnt], 0,

Contour Approximation Method

There are two approximation methods-cv.CHAIN_APPROX_NONE, The first figure shows this method. In this all the boundary points are there.

The other one is cv.CHAIN_APPROX_SIMPLE. The second figure shows this method. It removes each and every redundant point and 4 points of the contour are left and saves memory.

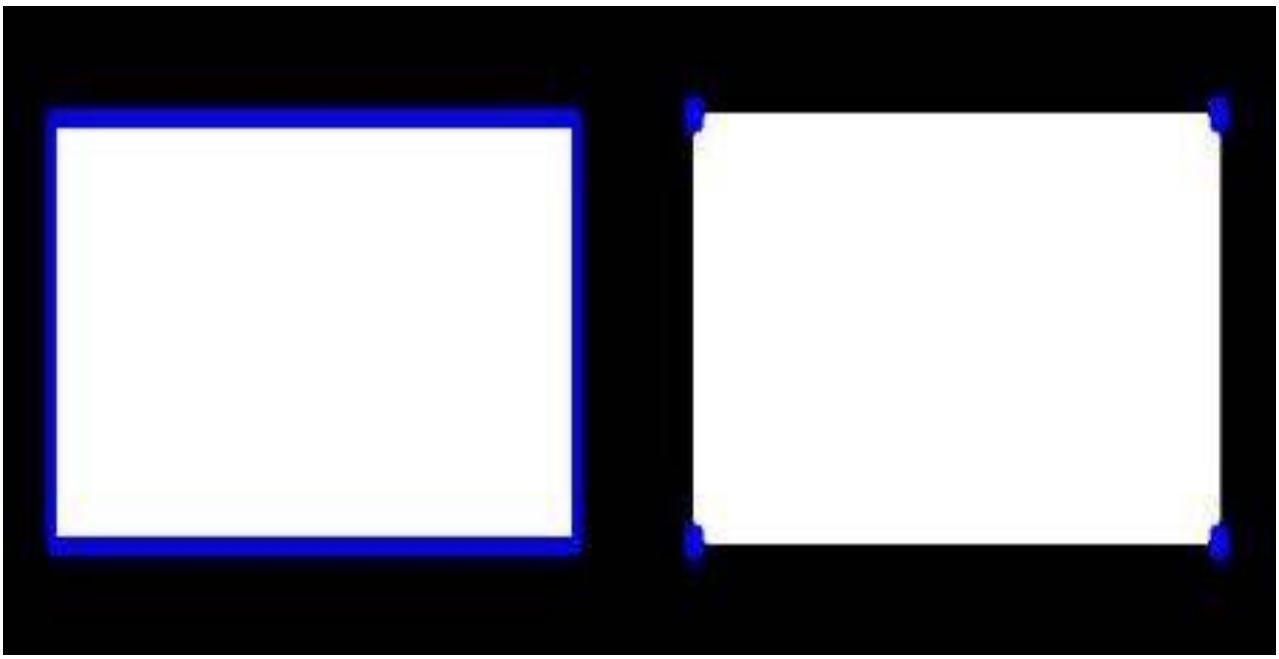

Figure 4.5: Difference between two types of contours

Contour Properties

- **Aspect Ratio**

  AspectRatio=Width/Height

- **Extent**

  Extent=ObjectArea/BoundingRectangleArea

- **Solidity**

  Solidity=ContourArea/ConvexHullArea

- **Equivalent Diameter**

  $$\text{EquivalentDiameter}=\left(\left(\frac{4\times\text{ContourArea}}{\text{pi}}\right)^{\frac{1}{2}}\right)$$

- **Maximum value, minimum value and their locations**

  This function gives all values required for this

  min_val, max_val, min_loc, max_loc = cv.minMaxLoc(imgray,mask = mask)

4.7 GrabCut Algorithm

It was given by Andrew Blake, C. Rother and V. Kolmogorov in Microsoft Research in Cambridge, UK. This is an algorithm which is used for foreground extraction with minimum interaction with the user.

First we will draw a rectangle around the object which we need to segment out. In our case we need to segment out the tailfin so the object is tailfin. Then this algorithm segment out the tailfin which we need iteratively and then gives the output. But in rare cases, the segmentation can go wrong or the tailfin may not come out clearly, like it get confused between foreground and background. In those cases we need to give some touch ups like we need to tell that this is foreground but you marked it as background or it is background you marked it as foreground.

Working of the algorithm

- First we select the rectangle by left clicking the mouse button and dragging it to make rectangle. The tailfin is taken in this rectangle and the outside of the rectangle is surely the background. We will specify which is background and which is foreground.
- Computer will do initial labelling. It gives names to the foreground and background pixels.
- The model which is used for the foreground and background isGaussian Mixture Model(GMM).
- This model learns and create the new pixel distribution. The pixels which are not known are labelled as either probable foreground or probable background which depends on other pixels which are surely background or foreground. It happens just like clustering.
- We make the graph from this pixel distribution. In the graph nodes are the pixels.

- The two nodes are added- the source node which is connected to foreground pixel and sink node which is connected to sink node.
- The weight of these two nodes can be calculated as the probability that the pixel is foreground or background. If there is more difference between pixel colors, then edge between them will be of less value.
- Then the graph is segmented by mincut algorithm. This algorithm will cut the graph into two parts- source node and sink node. These parts are cut with minimum cost function. The cost function is defined as the sum of all weights of all the edges that are cut by this algorithm.
- After cutting the edges, all the pixels those are connected to Source node= foreground and those are connected to Sink node = background.
- This is repeated iteratively till the classification converges.

cv.grabCut() is the function use to implement this algorithm.

Its arguments are-

- Input image
- mask - It is a mask image. When we pass the values 0,1,2,3 to image it means whether it is foreground or background or probable foreground or probable background.
- Coordinates of rectangle (x,w,w,h) in which there should be the foreground image
- bdgModel, fgdModel - These are arrays which can be used.First we have to create two np.float64 zero arrays whose size should be (1,65).
- Count - Number of iterations the algorithm should run.
- mode – there are two modes cv.GC_INIT_WITH_RECT or cv.GC_INIT_WITH_MASK. They can be combined also.

4.8 Hough Line Transform

The Hough Transform is used to identify any shape if that shape is in a mathematical form. If the shape is broken from between the also it can be detected.

If line is passing from below of the origin then P.D=+ve $\theta<180$, if line is going above the origin thanP.D=-ve $\theta<180$, vertical lines $\theta=0$ and horizontal lines $\theta=90$. The lines can be represented as, (P.D,θ). Firstly the 2D array is created to store the values which are all set to 0 by default. Rows are P.D and columns are θ.

4.9 Shi Tomasi Algorithm

One early endeavor to discover these corners was finished by Chris Harris and Mike Stephens in their paper A Joined Corner and Edge Indicator in 1988, so now it is called Harris Corner Identifier. He took this straightforward plan to a scientific structure.

The score is calculated by eigenvalues- $\lambda1$,$\lambda2$.

The R value for Harris corner detector is-

$R=\lambda1\lambda2-k(\lambda1+\lambda2)^2$

OpenCV has the capacity cv.cornerHarris() for this reason. Its contentions are :

- img - Information picture, it ought to be grayscale and float32 type.
- blockSize - It is the extent of neighborhood considered for corner location
- ksize - Gap parameter of Sobel subsidiary utilized.
- k - Harris finder free parameter in the condition.

Corner with SubPixel Exactness

Here and there, you may need to discover the corners with most extreme precision. OpenCV accompanies a capacity cv.cornerSubPix() which further refines the corners recognized with sub-pixel exactness. The following is a precedent. Obviously, we have to discover the harris corners first. At that point we pass the centroids of these corners (There might be a cluster of pixels at a corner, we take their centroid) to refine them. Harris corners are set apart in red pixels and refined corners are set apart in green pixels. For this capacity, we need to characterize the criteria when to stop the cycle. We stop it

after a predefined number of cycle or a specific exactness is accomplished, whichever happens first. We likewise need to characterize the extent of neighborhood it would scan for corners.

The Shi-Tomasi corner detector is derived from Harris corner detector. However, one change is made in this detector which is much better than the other. Even if the Harris corner detector fails this algorithm gives better results.

There is a corner selection criteria. The R score is calculated for each pixel. If the value of R is above threshold value, the pixel is said to be a corner.

Shi and Tomasi suggested that only λ1,λ2 should be used to check if the pixel is a corner or not. The score is calculated by eigenvalues- λ1,λ2.

The R value for **Shi Tomasi** corner detector is- R=min(λ1,λ2)

If the value is greater than a threshold value it is considered as a corner.

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

img = cv2.imread(r"C:\Users\kanav\Desktop\Edged Output.png")
gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)


corners = cv2.goodFeaturesToTrack(gray,20,0.4,30)
corners = np.int0(corners)

for i in corners:
    x,y = i.ravel()
    cv2.circle(img,(x,y),3,255,-1)
corners = cv2.goodFeaturesToTrack(gray,20,0.4,20)
corners = np.int0(corners)
for i in corners:
    x,y = i.ravel()
    cv2.circle(img,(x,y),3,255,-1)

plt.imshow(img),plt.show()

```

Figure 4.6: Code for implementation of Shi Tomasi.

Chapter 5: Results & Conclusions

The input will be an image of the airplane.

Figure 5.1: Original image.

This is the first screen we will get from the code which is a black screen.

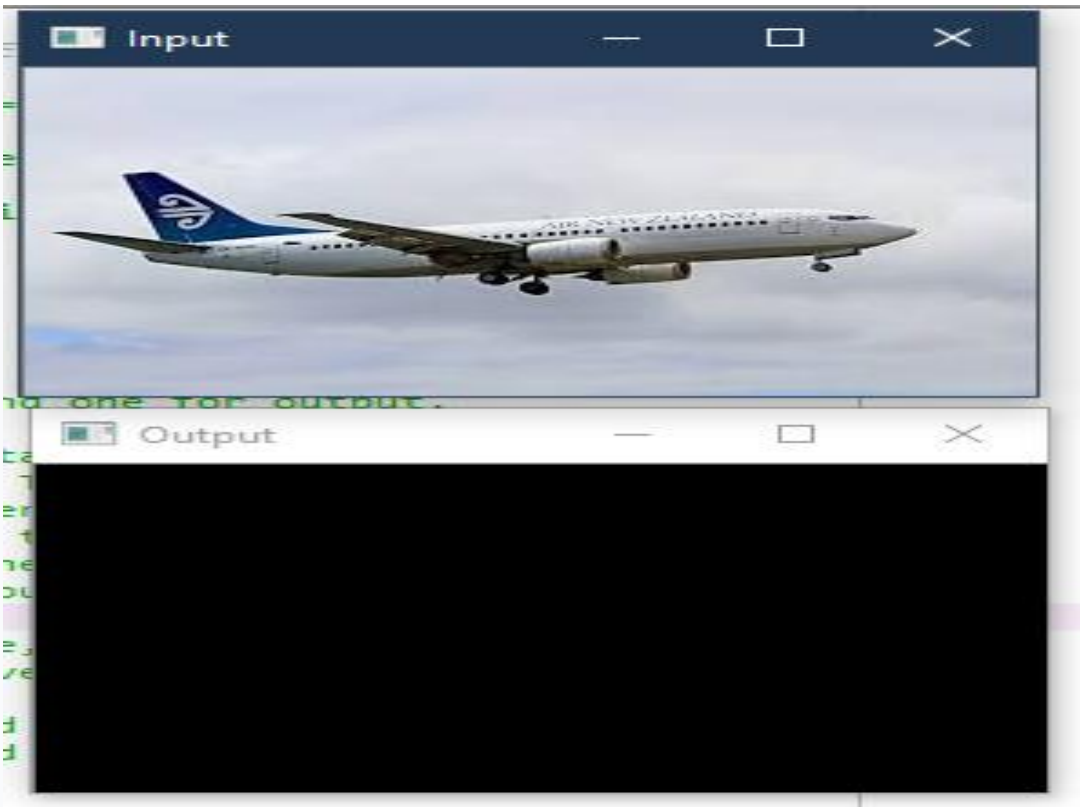

Figure 5.2: First output sceen.

Then by pressing left mouse button we make a rectangle around the tailfin we require.

To mark the areas of sure foreground press the number 1 and then use the mouse pointer to mark the areas of foreground. It will turn white in the input which shows it is foreground.

Press n to update the segmentation.

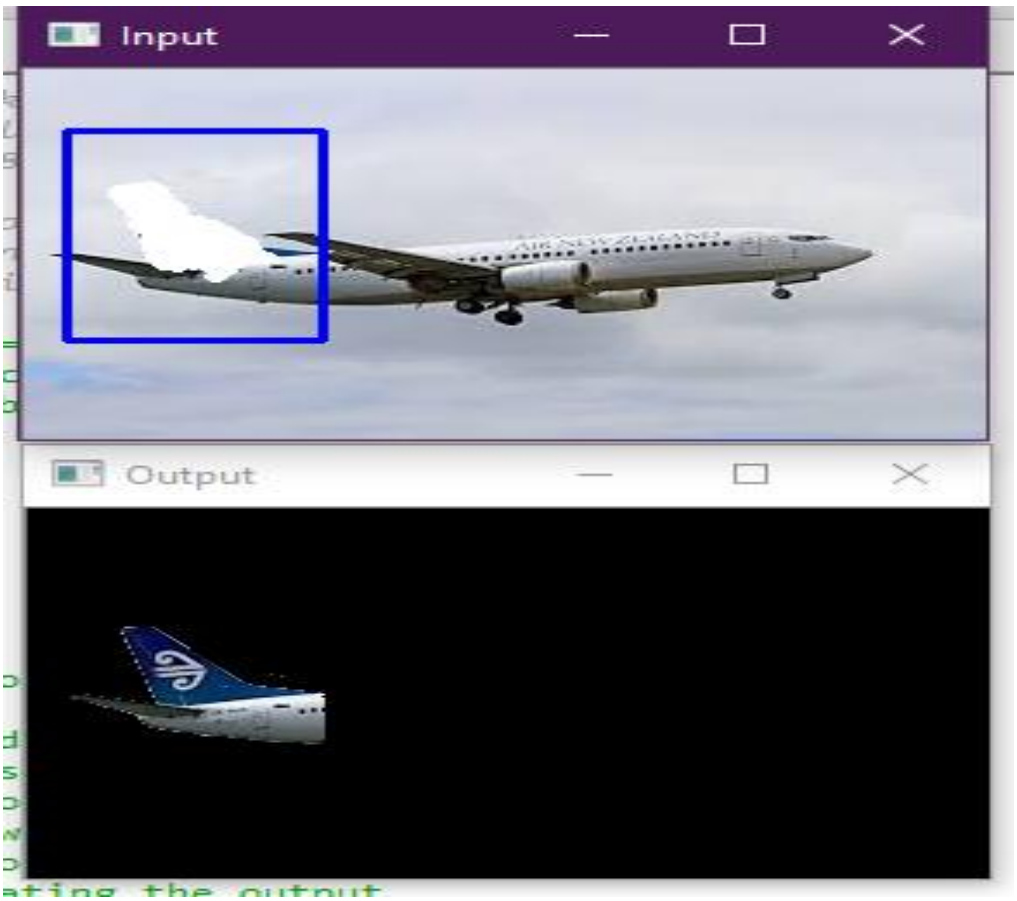

Figure 5.3: The foreground screen of tailfin which we require.

To mark the areas of sure background press the number 0 and then use the mouse pointer to mark the areas of background. It will turn black in the input which shows it is background.

Press n to update the segmentation.

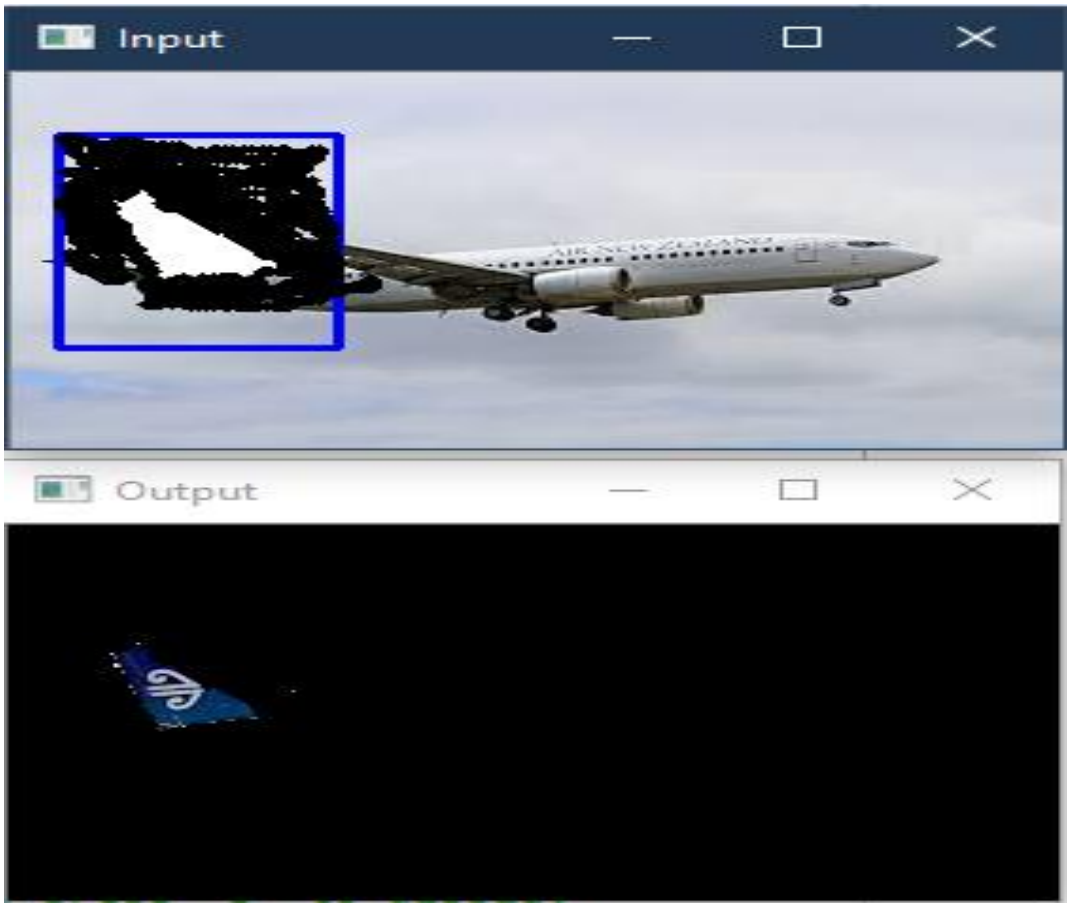

Figure 5.4: The areas of background which we do not require are painted black.

Now when the tailfin is segmented out then we will press s to display the output of Canny edge cut algorithm and the whiteboard.

Save the output of the image.

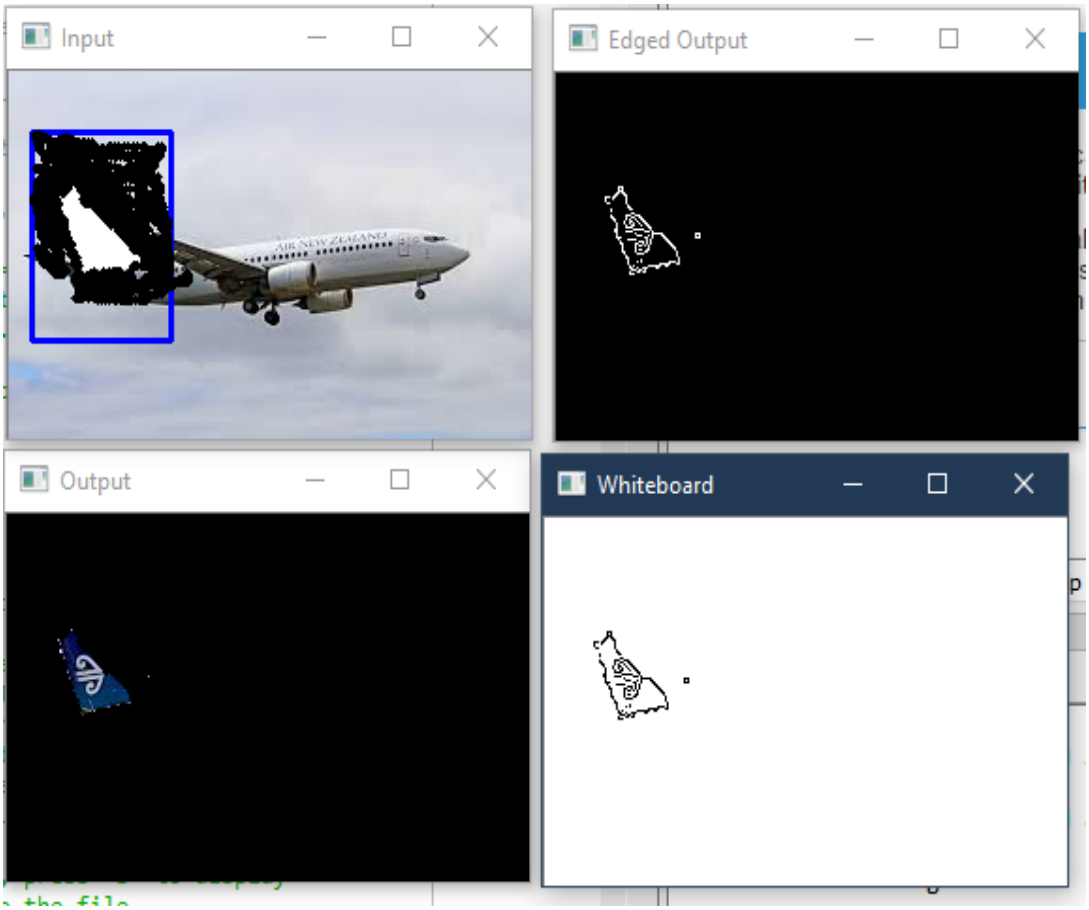

Figure 5.5: Canny edge detection is applied and we get the output.

Press esc key to exit.

After that Shi Tomasi Algorithm is applied to this output and the corners of the tailfin are detected.

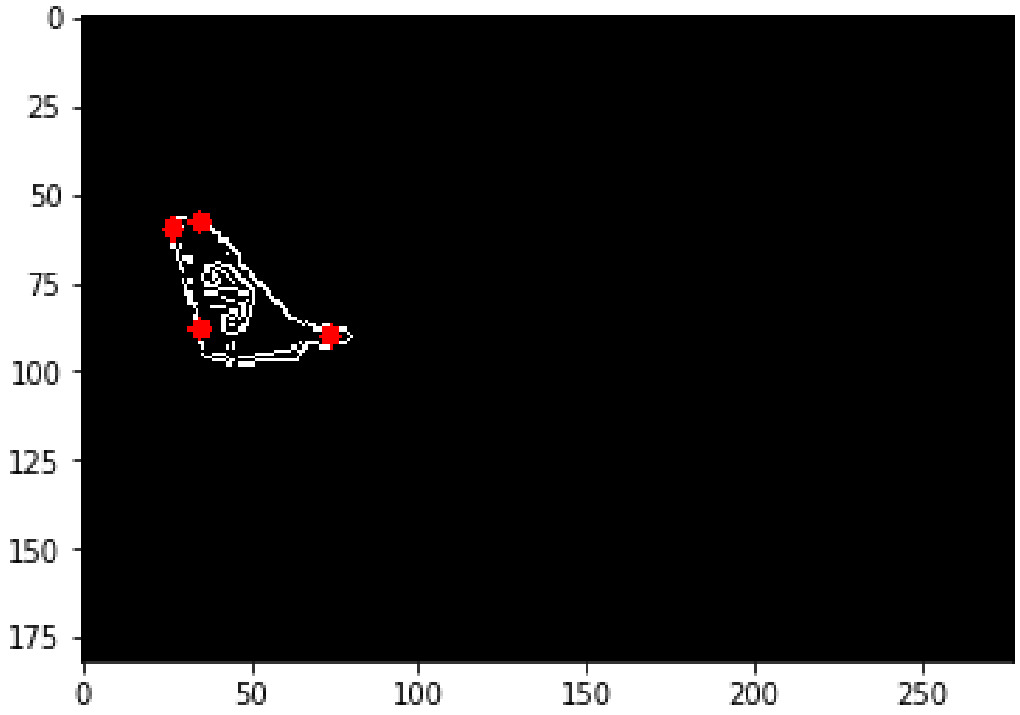

Figure 5.6: We get the 4 corners by applying Shi Tomasi algorithm.

5.4 Conclusions and Future Scope

Object location is a key capacity for most PC and robot vision framework. Albeit extraordinary advancement has been seen in the most recent years, and some current strategies are presently part of numerous shopper hardware (e.g., face discovery for auto-center in cell phones) or have been incorporated in colleague driving advances, we are still a long way from accomplishing human-level execution, specifically as far as open-world learning. It ought to be noticed that object identification has not been utilized much

in numerous zones where it could be of extraordinary assistance. As versatile robots, and all in all self-governing machines, are beginning to be all the more generally sent (e.g., quad-copters, rambles and before long administration robots), the need of item discovery frameworks is increasing more significance. At long last, we have to think about that we will require object discovery frameworks for nano-robots or for robots that will investigate zones that have not been seen by people, for example, profundity parts of the ocean or different planets, and the location frameworks should figure out how to new article classes as they are experienced. In such cases, an ongoing open-world learning capacity will be basic.

We implemented the segmentation algorithms to segment out the tailfin. As shown in the pictures, the tailfin of the airplane was easily recognizable. After that we apply Canny edge detection algorithm to get the edges of the tailfin and from this output we calculate the corners of the tail fin using Shi Tomasi algorithm.

Then we have to join all the corners that we get from Shi Tomasi algorithm. After that the Hough Line transformation can be used to clear the lines so that it is not distorted. Then we can use cosine rule to find the angles.

This project can be further used for finding the orientation of different objects. The wave points can be found out for the drone moving according to the orientation of different objects.

References

[1] RAatnakirti Roy, Anirban Sarkar and Suvamoy changder, Evaluating Image Steganography Tachniques, Senior Member, IEEE.

[2] Sumeet Kaur and R.K.Bansal, Steganography and classification of steganography techniques, IEEE Trans. Intelligent Transportation Systems, vol. 7, no. 3,2006, pp. 377–392.

[3] N.F. Johnson, Z. Durie, S. Jajodia, Information hiding: Steganography and watermarking - attacks and countermeasures in , Kluwer Academic Publishers, 2000.

[4] J. Fridrich, R. Du,, L. Meng,, Steganalysis of LSB Encoding in Color Images, Proc.

IEEE Int'l Conf. Multimedia and Expo, 2000.

[5] Rafael C. Gonzalez and Richard E. Woods, Digital Image Processing, 3rd edition, 200