Hands-free image cropping using sensors

B.Tech Project-I (CS47007) report submitted to

Indian Institute of Technology Kharagpur

in partial fulfilment for the award of the degree of

Bachelor of Technology

in

Computer Science and Engineering

by

Suchintan Pati

(18CS10064)

Under the supervision of

Professor Partha Pratim Das

Department of Computer Science and Engineering

Indian Institute of Technology Kharagpur

Autumn Semester, 2021-22

November 13, 2021

DECLARATION

I certify that

(a) The work contained in this report has been done by me under the guidance of my supervisor.

(b) The work has not been submitted to any other Institute for any degree or diploma.

(c) I have conformed to the norms and guidelines given in the Ethical Code of Conduct of the Institute.

(d) Whenever I have used materials (data, theoretical analysis, figures, and text) from other sources, I have given due credit to them by citing them in the text of the thesis and giving their details in the references. Further, I have taken permission from the copyright owners of the sources, whenever necessary.

Date: November 13, 2021 (Suchintan Pati)
Place: Kharagpur (18CS10064)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR

KHARAGPUR - 721302, INDIA

CERTIFICATE

This is to certify that the project report entitled "**Hands-free image cropping using sensors**" submitted by **Suchintan Pati** (Roll No. 18CS10064) to Indian Institute of Technology Kharagpur towards partial fulfilment of requirements for the award of degree of Bachelor of Technology in Computer Science and Engineering is a record of bona fide work carried out by him under my supervision and guidance during Autumn Semester, 2021-22.

Date: November 13, 2021

Place: Kharagpur

Professor Partha Pratim Das
Department of Computer Science and Engineering
Indian Institute of Technology Kharagpur
Kharagpur - 721302, India

Abstract

Name of the student: **Suchintan Pati** Roll No: **18CS10064**

Degree for which submitted: **Bachelor of Technology**

Department: **Department of Computer Science and Engineering**

Thesis title: **Hands-free image cropping using sensors**

Thesis supervisor: **Professor Partha Pratim Das**

Month and year of thesis submission: **November 13, 2021**

Hand gesture recognition has been an active area of research due to growing interests in human computer interaction. In this project, we solve the task of image cropping via hand tracking and detection by leveraging some state-of-the-art techniques. An efficient and robust method of hand detection and hand tracking is applied to define gestures and crop an image.

Acknowledgements

I would like to express my sincere thanks to my advisor Prof.Partha Pratim Das. He was always supportive and approachable. I would also express my gratitude to Himadri Bhuyan for helping me with a lot of valuable discussions. I express my profound thanks towards all faculty of the Computer Science and Engineering department at Indian Institute of Technology, who build strong foundations for research through various courses. Finally, I want to thank my family members and friends for their continuous support and love.

Contents

List of Figures

Chapter 1

Introduction

Human Computer Interaction has been a growing field of research. With the growing popularity of Augmented and Virtual Reality (AR/VR), gesture recognition and hand tracking play an important role. Through this project, we aim to explore these research areas by performing the task of hands-free image cropping. A robust and efficient method is discussed, which can be used in a wider range of hands-free applications.

1.1 Objective

The aim of the BTP is to be able to crop an image through a series of hands-free gestures via the aid of sensors/cameras. These gestures include tracking gesture that is responsible for drawing a bounding area over the intended image to be cropped and control gestures that assist the user in achieving the previous task.

1.2 Related Work

In this section, some relevant works in the field of hand gesture recognition have been listed.

Regional Attention with Architecture-Rebuilt 3D Network for RGB-D Gesture Recognition (Zhou et al., 2021) This paper proposes a regional attention with architecture-rebuilt 3D network (RAAR3DNet) for gesture recognition from RGB-D input (can be collected from depth cameras) .The fixed Inception modules in the backbone model of I3D (Inflated 3D network) is replaced with automatically rebuilt structure through the network via Neural Architecture Search (NAS), owing to the different shape and representation ability of features in the early, middle, and late stages of the network. This enables the network to capture different levels of feature representations at different layers more adaptively. Apart from replacing the standard Inception modules, additional regional attention modules are also added to the network at the three stages of the network. These modules are called Dynamic Static Attention (DSA). This module further consists of two sub modules, namely Dynamic attention sub-module and Static attention sub-module. The Dynamic attention sub-module is affiliated with the effective motion information among frames and the Static attention sub-module is guided by a heatmap which is related to the location of keypoints in hands/arms regions. This network utilises a two-stream configuration where each stream is trained separately for RGB and depth inputs respectively.

Online Detection and Classification of Dynamic Hand Gestures with Recurrent 3D Convolutional Neural Networks(Molchanov et al., 2016) In this paper a recurrent three-dimensional convolutional neural network is proposed that performs simultaneous detection and classification of dynamic hand gestures from multi-modal data (for eg. colour and depth). This model works online in order

to avoid any noticeable lag between performing a gesture and its classification. In fact, it provides a negative lag, that is, classification is done before the gesture is even finished. First, the sample video is fragmented to a certain number of clips. There are two modes of classification: offline and online. In offline classification negative log-likelihood is taken as the cost function, where the probability is given by the average of all the outputs post softmax layer. For online classification a special cost function called connectionist temporal classification is used instead. While predicting again average of all softmax layers is taken for offline video and only clip wise probabilities is taken for online sequences. For different input modalities like colour and depth, the network is trained separately and their probabilities are averaged.

Real Time Gesture Recognition System for Interaction in Dynamic Environment (Rautaray and Agrawal, 2012) The method used here is quite different from the above methods. Instead of using CNNs for feature extraction, which are computationally expensive and memory intensive, hand crafted features are used for gesture classification. After the image of the hand is captured by the camera, the parameters are limited to the minimum possible level by removing unnecessary information. Then the haar cascade classifier is responsible for locating hand position and classifying gestures. Hand tracking is done via camshift technique with shifting the region of interest with average shift in the object of interest i.e hands. As the hand is tracked a contour is mapped with the corresponding hand which further extracts a corresponding convex hull. The recognition has been done through modeling of the hand by mapping it to the number of defects formed in it. Afterwards the system tracks the number of defects that have been generated by the hand and maps it to a meaningful command. The dynamic user interface was designed using the image processing techniques which were implemented in C++ with the use of OpenCV Library. The same gesture vocabulary can be used for different applications like controlling games, browsing images, etc.

Robust Hand Gesture Recognition Based on Finger-Earth Mover's Distance with a Commodity Depth Camera(Ren et al., 2011) This paper focuses on building a robust hand gesture recognition system using the Kinect sensor. In order to handle the noisy hand shape obtained from the Kinect sensor, a distance metric for hand dissimilarity measure, called Finger-Earth Mover's Distance (FEMD) is proposed. This metric only matches fingers while not the whole hand shape. Thus it can better distinguish hand gestures from slight differences. Firstly, hand segmentation is done via gathering the input depth data, given that the hand is the foremost object for the sensor. The user needs to wear a black band on his/her wrist to facilitate a more precise hand shape to be detected. After detection, the hand shape is represented in the form of a time-series curve. This records the relative distance between each contour vertex to a center point. Each finger generally corresponds to a peak in this curve. FEMD is modification of EMD (Earth Mover's Distance), which is used to measure distance between signatures and histograms. The input hand is then classified as the class with minimum dissimilarity distance, given by this FEMD calculation.

The latter two approaches discussed above are not robust when subject to a variety of gestures and environments. The FEMD method only deals with static gestures and doesn't address the hand-tracking problem. Although the former two methods((Zhou et al., 2021) and (Molchanov et al., 2016)) are robust, training the models on a custom dataset is difficult. Apart from that these models are resource intensive, which implies they are not suitable to be deployed on low-end devices. Thus an approach where limited resources restriction without compromising on robustness is needed. The *MediaPipe Hands model* by Google is one such model that has been applied in this BTP. This model is described in detail in the next section.

Chapter 2

Background Study

This chapter contains some details of algorithms and models that were used in the implementation.

2.1 Image Processing

2.1.1 Matrix Representation

Images are stored in matrices, whose dimensions and values are determined by the type of information represented by each pixel. For RGB images, each pixel requires a tuple of numbers to represent the information. Therefore a 3 dimensional matrix is needed to represent RGB image. A pixel in an RGB image consists of three stream of colours, where each stream consists of 8 bits. Therefore value of a stream can lie between 0 and 255 (both inclusive). Let *R(x,y)*, *G(x,y)* and *B(x,y)* denote the red, green and blue intensities of pixel at *(x,y)* coordinate respectively. Then we have: $(0 \leq R(x,y) < 256)$, $(0 \leq G(x,y) < 256)\, and\, (0 \leq B(x,y) < 256)$. This is shown in the Figure 2.1[1]In the case of grayscale images, each pixel conveys information about the intensity of light. Hence a 2 dimensional matrix can represent a grayscale

image, where let each pixel at coordinate *(x,y)* be represented by intensity *Gray(x,y)*. Similarly binary images are also represented by 2 dimensional matrices containing either 0 or 1 values.

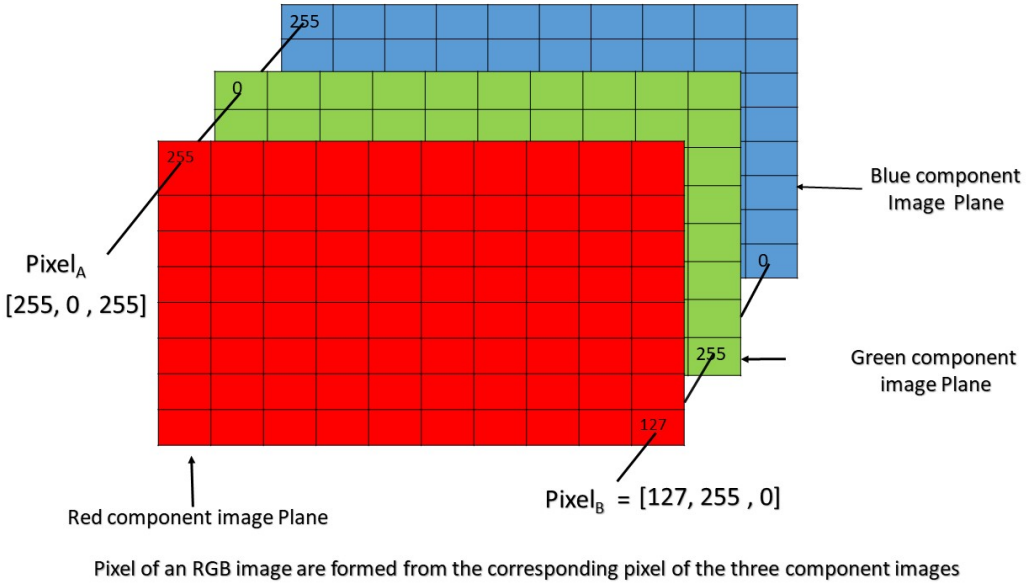


FIGURE 2.1: Matrix Representation of RGB image

2.1.2 Color space conversion

RGB to Grayscale There are two methods to convert an image from Grayscale to RGB, namely: average method and weighted method. For this implementation, the weighted method (also called luminosity method) is used, where the red, green and blue values are weighted according to their wavelengths:

$$Gray(x, y) = 0.299 * R(x, y) + 0.587 * G(x, y) + 0.114 * B(x, y)$$

Grayscale to RGB Here the grayscale value is assigned to all three color channels:

$$R(x, y) = G(x, y) = B(x, y) = Gray(x, y)$$

[1]Image source: https://media.geeksforgeeks.org/wp-content/uploads/Pixel.jpg

Grayscale to Binary Grayscale images are converted to binary based on the intensity values of a pixel with respect to a threshold value. If the pixel value exceeds the threshold then it is given a value of 1 and vice versa. An inverse binary conversion does the opposite.

2.2 MediaPipe Hands

MediaPipe Hands: On-device Real-time Hand Tracking by *Google Research(Zhang et al., 2020)* proposes a novel highly optimized method to detect and track hands. The hand-landmark model provide key-points that identify specific parts of hands for example, the joints and tips. In the following sections the various models deployed and their working principles are explained in brief.

2.2.1 Architecture

The solution consists of two models working together: a palm detector model and a hand landmark model.

- The palm detector module, also called as BlazePalm, takes input as an image frame and returns a bounding box containing the hand
- The hand landmark module takes the bounding box as input and performs precise land-mark localization of 21 2.5D coordinates inside the detected hand regions via regression. Some examples of these points are: wrist, index finger tip, index finger dip, etc.

The hand tracking pipeline consisting of the above two models can be built as a directed graph of modular components with the help of MediaPipe(Lugaresi et al., 2019) (see Appendix A for more details). The MediaPipe graph shown in the Figure

2.2 consists of the palm detector and the hand landmark model. This type of pipeline helps in the following optimization: the palm detector module runs only for the first frame and when the hand is not detected. This saves a significant amount of computation as for the majority of the frames, only the hand landmark module, which is less computationally heavy, is active. The latter module also outputs a scalar denoting the confidence of hand presence in the input cropped image. If this confidence falls below an adjustable threshold, the palm detector module is invoked. The MediaPipe provides additional Calculators that aid the above models through data transformation, media processing, cropping etc. Some more details about the palm detector and hand landmark models are provided in the next section.

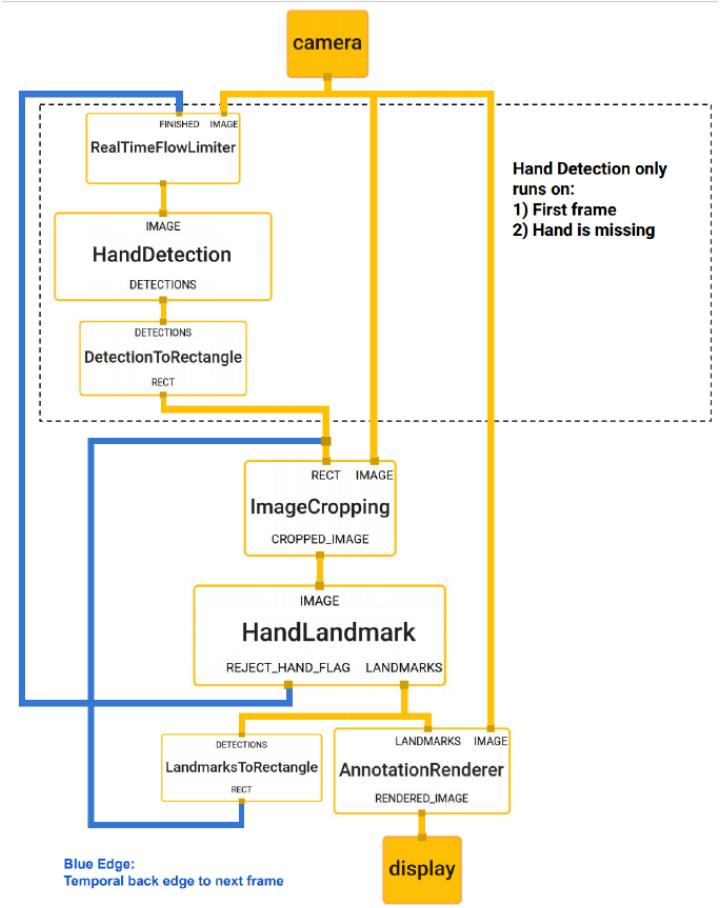


FIGURE 2.2: MediaPipe Graph

2.2.2 BlazePalm model

Detecting hands from a variety of hand sizes and occluded and self-occluded hands is a difficult task. A model similar to BlazeFace(Bazarevsky et al., 2019) is applied in this solution. In order to simplify hand detection, first, a palm detector is used that estimates bounding boxes of rigid objects like palms instead of hands with articulated fingers. Then an encoder-decoder feature extractor similar to FPN(Lin et al., 2017) for a larger scene-context awareness. The focal loss(Lin et al., 2018) is the loss function that is used.

2.2.3 Hand Land mark model

This model (Figure 2.3) performs a precise land mark localisation of 21 2.5D coordinates inside the cropped region. . The model produces the three outputs:

- 21 hand landmarks, like tips and joints(see Figure 3.1), consisting of x, y, and relative depth. These coordinates are obtained by performing a regression on the the received cropped input frame. The model learns a consistent internal hand-pose representation which is robust to self occlusion and partial visibility. The 2D coordinates are learnt from both real world and synthetic images, whereas relative depth w.r.t. the wrist point is learnt only through synthetic images.
- A hand flag indicating the probability of hand presence in the input image. If the score is lower than a threshold, then the detector is invoked. This score is calculated taking a cumulative score of individual landmarks.
- A binary classification of handedness, i.e. whether it is left or right hand. This is done by simply adding a layer after the feature extractor.

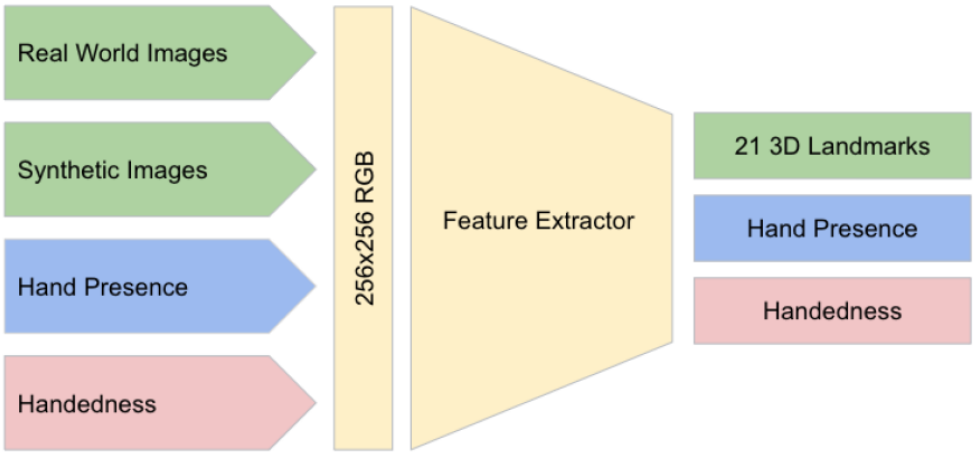


FIGURE 2.3: Hand Landmark model architecture

2.2.4 Dataset

The solution was trained on the following different datasets:

- In-the-wild dataset, that consists of hand images with a variety of lighting conditions, hand appearances.
- In-house collected gesture dataset, consisting of hand images taken from a variety of angles and covers almost all types of gestures possible.
- Synthetic Dataset, consists of rendered high quality synthetic hand model over diverse backgrounds and mapped to corresponding 3D coordinates.

Chapter 3

Approach

In this chapter, implementation details of the project is discussed followed by some key observations and future scope of this project.

3.1 Implementation

The task of hands-free image cropping was achieved with the help of MediaPipe Hands model and various image processing techniques. An overview of the steps taken is provided, where each step is described in subsequent sections.

1. An input frame (from webcam) is captured from which the keypoints are obtained from the hand landmark model.
2. Identify gestures from the keypoints obtained.
3. Perform the cropping of image after tracking the movement of index finger.

There are three images that are stored as matrices that will be operated upon to get the desired output:

- *Cam_Image:* Image frame captured by the camera.
- *Crop_Image:* The image that the user is going to crop.
- *Image_Canvas:* A utility matrix, that assists in executing the actions of gestures.

3.1.1 Keypoints collection

The MediaPipe Hands solution, discussed in the previous chapter, is applied on the input frame to obtain the 21 x and y landmark coordinates. These points are also displayed on the screen. The 21 landmarks contain the coordinates of all the joints and tips of all fingers. The thumb is represented by its tip, and joints namely: Carpometacarpal (CMC), Metacarpophalangeal (MCP) and Interphalangeal (IP) joint. The remaining four fingers are characterised by their tip and Distal Interphalangeal joint (DIP), Proximal Interphalangeal joint (PIP) and Metacarpophalangeal joint (MCP). These joints are depicted in the Figure 3.1.

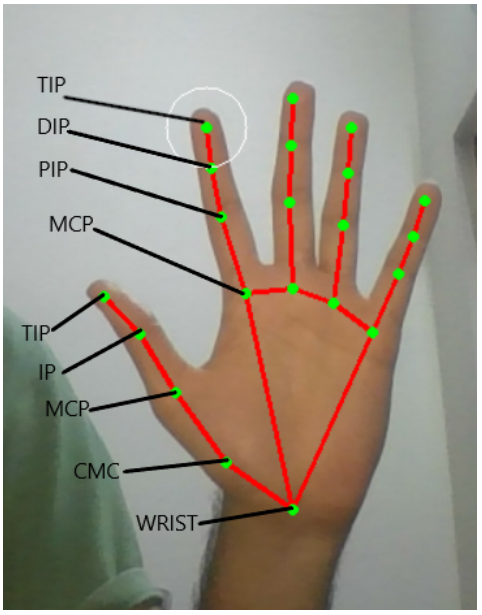


FIGURE 3.1: Hand keypoints

3.1.2 Gestures

Using the stored coordinates of each finger the following gestures have been defined: Draw, Clear screen and Crop. The gestures are recognised based on the relative position of these finger joints. A finger is said to be up if the y coordinate of the PIP is lower than the y coordinate of the tip, i.e.

if $TIP[i].y > PIP[i].y$ **then** ▷ i represents the finger index
  $FingerUp[i] \leftarrow True$
else
  $FingerUp[i] \leftarrow False$
end if

The same is done for the thumb, but the y coordinate of IP is compared with that of the tip. After identifying the gesture, the program reaches a particular state and performs the corresponding action.

3.1.2.1 Draw Gesture

This gesture is triggered when only the index finger is shown, i.e. when only *FingerUp[1]* is true and others are false. The coordinate of the index finger tip in the current frame is added in a list. Let *points[]* be the list containing the coordinates of index finger tip from all the frames since the beginning. Hence, as the user moves his index finger freely in front of the camera, the corresponding shape consisting of points with x-y coordinates is stored. A line is drawn dynamically between the previous and current position of the index finger tip for each frame. These lines are only for a view purpose that help the user keep a track of the figure drawn till now. This is done by:

1. Drawing the line on the Image_Canvas first, followed by doing a color space conversion from RGB to grayscale.

2. An inverse binary conversion is applied on the grayscale image, post which the binary image is converted back to RGB. Let this image be X.

3. A bit-wise and operation is done on Crop_Image with X to obtain an image Y. This is followed by a bit-wise or operation between Y and Image_Canvas to preserve the color of line. This final image is then displayed.

The user can now see the line drawn to appear on the Crop_Image (See Figure 3.2).

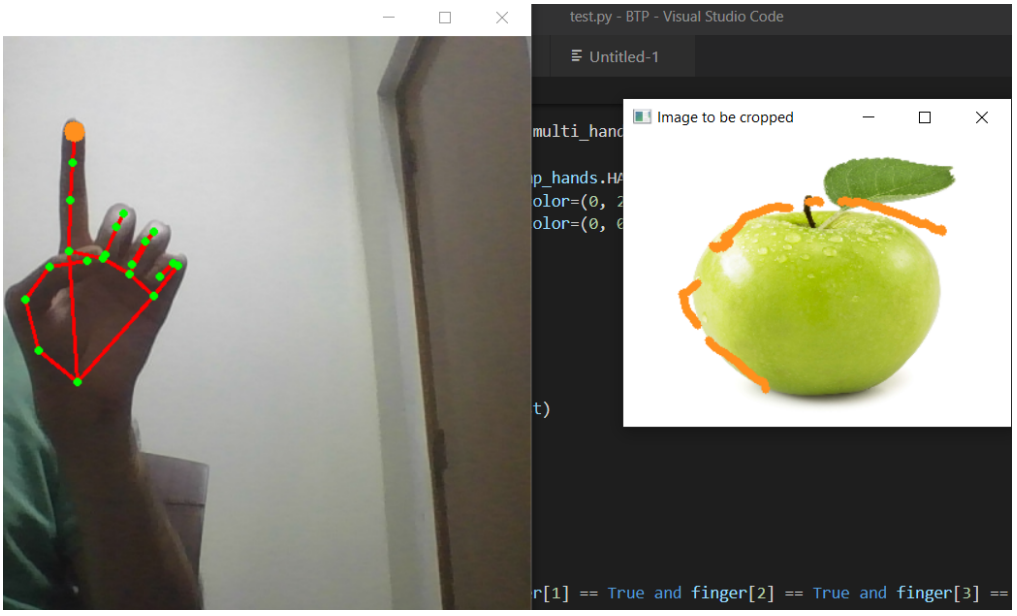


FIGURE 3.2: Draw Gesture

3.1.2.2 Clear Screen Gesture

When the user shows all five fingers, i.e. when $(FingerUp[i] == True)\ \forall i \in [0, 4]$, then a clear screen gesture is invoked, which as the name suggests clears the screen (Figure 3.3) and removes all the points that were stored in the list *points[]* in the draw gesture. This is done by filling the Image_Canvas with black (same as re-initialising the matrix with zeroes), followed by performing the steps 2-3 mentioned in Draw gesture. The user will see that all the lines drawn on top of the Crop_Image will disappear. The state of the program reaches to start.

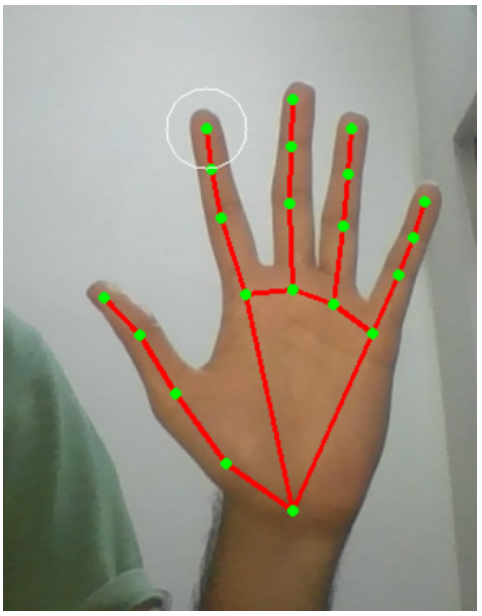


FIGURE 3.3: Clear Screen Gesture

3.1.2.3 Crop Gesture

The cropping action is invoked when the user shows the index and the middle finger together (victory sign!), i.e. *FingerUp[1]* and *FingerUp[2]* are true and remaining are false (Figure 3.4). When the cropping action is invoked:

1. A contour is drawn on a mask, which is a matrix initialised with zeroes, by joining the points stored in *points[]* during the drawing phase.

2. A closed figure is obtained by joining the first and the last point. This figure is filled with a color. Now all the pixels in the mask outside this figure are zeroes.

3. The cropped image is then extracted by performing bit-wise and operation on the Crop_Image with itself and the mask. A bounding rectangle is also obtained based on the figure formed via the list *points[]* (by taking the topmost, bottom-most, leftmost and rightmost points) . The cropped image is then resized to the dimensions of this rectangle.

FIGURE 3.4: Crop Gesture

The cropped image now displayed on a separate window (Figure 3.5).

(a) Original Image (b) Cropped Image

FIGURE 3.5: Original and Cropped Image

3.1.3 Handling noise

There were many observed jitters during the drawing phase due to variety of factors during the detection and the tracking phase. These sudden movements caused

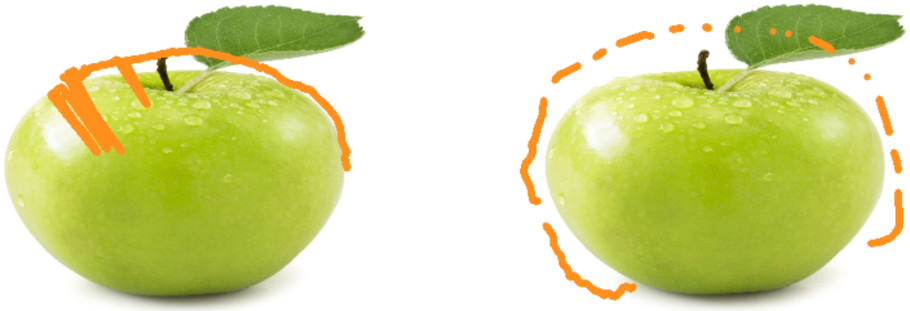
erratic drawing of points leading to inaccuracy. This was avoided by recording only points that were close to each other, where a distance threshold determines closeness. Algorithmically:

if $distance(x_{prev}, y_{prev}, x_{curr}, y_{curr}) \leq threshold$ **then**

(x_{curr}, y_{curr}) *appended to* $points[\,]$

end if

Thus, if one moves the index finger very quickly or there are sudden jitters due to imperfect detection, the corresponding outlier point(s) will be ignored and the drawing phase continues. This resulted in more separated points, where separation depends on the speed with which the user moves his hand. The same is depicted in Figure 3.6.



(a) Before (b) After

FIGURE 3.6: Improvement in drawing

3.2 Conclusion and Future Scope

Here are some key observations and limitations on this approach:

1. The field of view of the webcam is quite low. This means the hand will easily move out of the frame, and the image to be cropped has to be resized to almost half the field of view of the webcam.

2. The external factors also play a major role in determining the detection and tracking confidence. For example, colour plays a huge factor: an orange or close to skin colour background can deter the performance. Hence there is some reduction in performance if the hand is in front of the face.

   Future Scope The project initially required the use of depth-sensing cameras. However, due to technical challenges, the sensor's didn't work (couldn't take input). The first point, i.e. the field of view issue can be resolved by using a more sophisticated sensor. The second point can be addressed with the use of depth information in addition to RGB will help to segregate the hand, which is closer to the camera, from the background to detect and track the hand.

   Therefore, the use of depth sensing sensors can improve the detection accuracy along with increasing the field of view, thereby allowing the user more room to work with.

Appendix A

Appendix

A.1 MediaPipe

MediaPipe: A Framework for Building Perception Pipelines (Lugaresi et al., 2019) provides a developer with an environment to run and improve their applications across various devices and platforms. In MediaPipe, all processing takes place with the context of a *Graph*. Each node in this graph is implemented as a *Calculator*. Each node may receive zero or more input streams and produce zero or more output streams. A source node has zero input stream and a sink node has zero output stream. An example in the context of this project: webcam input is the source node and Display is the sink node. Majority of execution time is spent inside these *Calculators*. These calculators can perform a variety of tasks like DetectionToRectangle, ImageCropping, FlowLimter ,etc. as shown in Figure 2.2. Flow limiters are responsible for detecting if the processing rate of frames is lower that the input rate, and drops packets accordingly. This type of architecture allows developers to focus on the algorithms of these *Calculators* while the MediaPipe framework provides an easy way to connect the nodes in graph. This allows parallelization as the *Calculators* can be executing simultaneously, therfore making a better use of the resources available.

Bibliography

Bazarevsky, V., Kartynnik, Y., Vakunov, A., Raveendran, K., and Grundmann, M. (2019). Blazeface: Sub-millisecond neural face detection on mobile gpus.

Lin, T.-Y., Dollar, P., Girshick, R., He, K., Hariharan, B., and Belongie, S. (2017). Feature pyramid networks for object detection. pages 936–944.

Lin, T.-Y., Goyal, P., Girshick, R., He, K., and Dollar, P. (2018). Focal loss for dense object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PP:1–1.

Lugaresi, C., Tang, J., Nash, H., McClanahan, C., Uboweja, E., Hays, M., Zhang, F., Chang, C.-L., Yong, M., Lee, J., Chang, W.-T., Hua, W., Georg, M., and Grundmann, M. (2019). Mediapipe: A framework for building perception pipelines.

Molchanov, P., Yang, X., Gupta, S., Kim, K., Tyree, S., and Kautz, J. (2016). Online detection and classification of dynamic hand gestures with recurrent 3d convolutional neural networks. pages 4207–4215.

Rautaray, S. and Agrawal, A. (2012). Real time gesture recognition system for interaction in dynamic environment. *Procedia Technology*, 4:595–599.

Ren, Z., Yuan, J., and Zhang, Z. (2011). Robust hand gesture recognition based on finger-earth mover's distance with a commodity depth camera. pages 1093–1096.

Zhang, F., Bazarevsky, V., Vakunov, A., Tkachenka, A., Sung, G., Chang, C.-L., and Grundmann, M. (2020). Mediapipe hands: On-device real-time hand tracking.

Zhou, B., Li, Y., and Wan, J. (2021). Regional attention with architecture-rebuilt 3d network for rgb-d gesture recognition.