Journal of Informatics
Electrical and Electronics Engineering, 2021,
Vol. 02, Iss. 01, S. No. 005, pp. 1-11
ISSN (Online): 2582-7006

Media Control Using Hand Gestures

Mrigank Singh[1], Sheenu Rizvi[2]

[1,2]Department of Computer Science and Engineering, Amity University Uttar Pradesh, Lucknow Campus, India
[1]mrigank2303239@gmail.com, [2]srizvi@amity.edu

How to cite this paper: M. Singh and S. Rizvi (2021) Media Control Using Hand Gestures. *Journal of Informatics Electrical and Electronics Engineering, Vol. 02, Iss. 01, S. No. 005, pp. 1-11, 2021.*

https://doi.org/10.54060/JIEEE/002.01.005

Received: 24/02/2021
Accepted: 17/03/2021
Published: 20/03/2021

Abstract

The corporate world today basically relies on presentations of ideas and statistics. In the board room, the presenters are highly conscious of depicting confidence in their presentation. This would entail accessibility and mobility to the presenter or media viewer. As the extent of Artificial Intelligence is increasing in all directions, I am utilizing its extreme capabilities to create a software that would help in accessibility and save time and money. This paper describes a software written in Python 3.8 and makes use of Python libraries like OpenCV and PyAutoGUI to receive input from the computer's Webcam and recognize gestures to control the PowerPoint Presentation and Portable Document Format (PDF) files or the Media Control. The user interface is built with the Python library PyQt5. This paper aims are to help people control their Presentations and Portable Document Format (PDF) files, and many other media through their hand gestures, without using a mouse or any other pointing device. The software would not require any other external hardware; hence it would not burn a hole in people's pockets.

Keywords

Artificial Intelligence, Deep Learning, Computer Vision, OpenCV, Python

1. Introduction

There have been numerous disclosures in the field of Computer Science to improve individuals' life and be more helpful, and Artificial Intelligence is the most astounding one. [1] Artificial Intelligence is getting increasingly more advanced as time passes. People are continually astounding themselves with the headway they have made around there; making machines think and work in an insightful way. It is concerned with the basic and most important aspects in our life i.e. philosophy, computer science, mathematics, linguistics, biology, neuron science, sociology etc. AI plays a very important role to exhibit intelligent behaviour, to learn, demonstrate and give advice to the user. [12]

Since I have been applying one of the numerous utilizations of Artificial Intelligence, I would like to introduce it in this section. The detection of real-world objects of interest, such as faces and people, poses challenging problems: these objects are difficult to model, there is significant variety in colour and texture, and the backgrounds against which the objects lie are unconstrained. [10] This field of Artificial Intelligence is called Object Detection, which would be the primary worry of my undertaking.

1.1. Artificial Intelligence

Right off the bat, we ought to comprehend what Artificial Intelligence is. Artificial Intelligence is the reproduction of human intelligence in machines with the goal that they can think and copy the human mind. [9] This term is related with any machine which can tackle issues or have attributes identified with human personalities. Usually, when a lot of people think about Artificial Intelligence, they consider Robots. This is because of the explanation that films and books everywhere in the world have consistently depicted robots to be human-like machines that are enormously cutting-edge and regularly make destruction on earth. In any case, this is a long way from truth or reality, and there is no ruin that we have to stress over.

Artificial Intelligence depends on the rule that human intelligence is characterized by a machine such that it can without much of a stretch execute errands like people; from the most basic positions to those that are unpredictable, in any event, for people themselves. The objectives of Artificial Intelligence incorporate learning, reasoning, and perception.

Artificial Intelligence is ceaselessly developing so various businesses are profited by it. Machines are wired utilizing a cross-disciplinary methodology situated in arithmetic, software engineering, semantics, brain research, and the sky is the limit from there. This cross-disciplinary methodology is the principle reason with respect to how Artificial Intelligence is being demonstrated useful in practically a wide range of businesses.

1.2. Characterization of Artificial Intelligence

Artificial Intelligence can be characterized as weak and strong Artificial Intelligence.

1.2.1 Weak Artificial Intelligence

Weak Artificial intelligence is designed to do one particular job, instead of many complex jobs at once. The examples of weak Artificial intelligence are: Amazon's Alexa, Apple's Siri or a machine playing chess. In all these examples the Artificial Intelligence has only one job, either to win or answer a question. A typical misguided judgment about weak artificial intelligence is that it's scarcely wise by any stretch of the imagination — more like counterfeit ineptitude than artificial intelligence. Yet, even the most intelligent appearing artificial intelligence of today are just weak artificial intelligence. As a general rule, at that point, tight or weak artificial intelligence is more similar to an astute trained professional. It's profoundly clever at finishing the particular undertakings it's customized to do.

1.2.2. Strong Artificial Intelligence

This kind of Artificial Intelligence is designed to behave even more like the human brain; hence their actions are more human-like. These are more complex and complicated systems. These systems are designed to handle situations in which they have to solve highly complex problems without the human's intervention. The examples of strong Artificial Intelligence are self-driving cars and hospital operating rooms. These systems need to be developed almost completely accurately, leaving only minor or no shortcomings whatsoever. For example: Included in numerous films, strong artificial intelligence acts more like a mind. It doesn't group, however utilizes bunching and relationship to deal with information. To put it plainly, it implies there is certifiably not a set solution to your watchwords. The capacity will impersonate the outcome, however for this situation, we aren't sure of the outcome. Like conversing with a human, you can accept what somebody would answer to an inquiry with, however you don't have the foggiest idea. For instance, a machine may hear "hello" and begin to connect that with the espresso producer turning on. In the event that the computer has the capacity, its hypothetically could hear "good day" and choose to turn on the espresso producer.

2. Computer Vision

Computer Vision can be characterized as a field of Artificial Intelligence that discloses how to remake, intrude, and comprehend

a 3-Dimensional scene from its 2-Dimensional pictures, regarding the properties of the structure present in the scene.[4] It manages demonstrating and imitating human vision utilizing computer programming and equipment. A straightforward similarity of Computer Vision can be the natural eye; the way where people see objects and the general climate and the mind responds in a proper way.[7] Computer vision helps scholars to analyse images and video to obtain necessary information, understand information on events or descriptions, and scenic pattern. [13] It resembles giving the machine eyes and encouraging its mind to work as per what it can see.
Computer Vision overlaps significantly with the following fields –

- **Image Processing** – This focuses on image handling or image manipulation.
- **Pattern Recognition** – This explains various techniques to identify and classify patterns.
- **Photogrammetry** – This is concerned with obtaining accurate measurements and dimensions from images.

Now, since we have discussed a lot about Computer Vision, it seems mainly like image processing. However, it is important to understand the difference between the two.[8]

Table 1. Differences between Computer Vision and Image Processing

Computer Vision	Image Processing
It is the construction of explicit, meaningful depictions of physical objects from their 2-dimensional image.	It deals with image-to-image transformation.
The output of computer vision is a portrayal or an interpretation of structures in 3-Dimensional scenes.	The input and output of image processing are both 2-Dimensional images.

3. Object Detection

The rapid development of computer industry production and computer intelligence, as well as the corresponding developments in computer-aided image analysis, has made industrial image processing to be a very important branch of scientific image processing. [11] Object detection is a computer vision procedure that helps in finding and distinguishing objects in a picture or a video. With the assistance of Object Detection, we can include the quantity of articles in a picture, recognize the sort of article appeared in the picture (for instance, a tree or a canine, and so on), or decide the exact area of any article with precise naming. This can be handily clarified by the accompanying model: Imagine a picture contains two felines and an individual. Object Detection encourages us to recognize and separate the felines and the individual without any problem. Object Detection is frequently mistaken for image recognition, thus first we have to comprehend the contrast between the two.

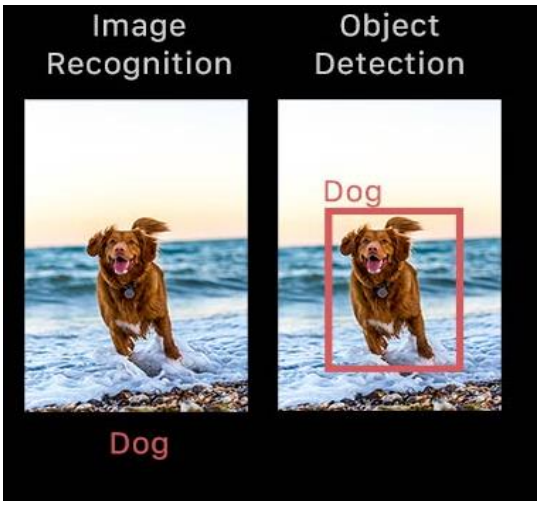

Figure 1. Difference between Image Recognition and Object Detection

Object detection, then again, draws a container around each canine and names the case "canine". The object detection model predicts where each object is, and what name ought to be applied to it. Along these lines, object detection gives more data about a picture and its substance, as opposed to simply acknowledgment.

3.1. Modes of Object Detection

Basically, there are two modes of object detection:

- Machine Learning based object detection
- Deep Learning based Object Detection.

In more conventional Machine Learning-based methodologies, Computer Vision procedures are utilized to take a gander at different highlights of a picture, for example, the shading histogram or edges to recognize gatherings of pixels that may have a place with an article. These highlights are then taken care of into a relapse model that predicts the area of the item alongside its mark. Then again, profound learning-based methodologies utilize convolutional neural networks (CNNs) to perform start to finish, unsupervised object detection, in which features don't need to be characterized and extracted independently.

3.2. Importance of Object detection

Object detection is breaking into a wide scope of businesses, with use cases going from individual security to profitability in the work environment. Facial detection is one type of it, which can be used as a safety effort to give just certain individuals access to an exceptionally arranged zone of an administration working, for instance. It tends to be utilized to check the quantity of individuals present inside a conference to consequently change other specialized apparatuses that will help smooth out the time devoted to that specific gathering. It can likewise be utilized inside a visual web search tool to help customers track down a particular thing they're on the chase for – Pinterest is one illustration of this, as the whole friendly and shopping stage is worked around this innovation.

4. OpenCV

OpenCV is the acronym for Open-Source Computer Vision Library. It is a library of functions which is used mainly for live Computer Vision in real-time. OpenCV was developed by Intel. It is a cross-platform library, which means that it can run on different

platforms easily. The library was originally written in C++ but it has bindings in many different languages such as Python, Java, Octave, MATLAB, etc. OpenCV provides a lot of functionality to the users when it comes to real-time Computer Vision.

The OpenCV library contains many capacities that help the catch, investigation, and control of visual data given to a computer by webcams, video records, or different kinds of gadgets. This contains many functions and algorithms for Motion tracking, Facial recognition, Object Detection, Segmentation and recognition and many other applications. [14] Straightforward capacities may be utilized to draw a line or other shape on a screen, while the further developed bits of the library contain calculations for identifying faces, following movement, and investigating shapes. Large numbers of this present library's calculations are identified with explicit employments of computer vision including item investigation, clinical imaging, advanced mechanics, facial and signal acknowledgment, and human-computer communication (HCI). As an open-source programming library, OpenCV can be utilized with not very many limitations in both business and specialist projects.

1.4.1. Why use OpenCV for computer vision?

OpenCV, or Open-Source Computer Vision library, began as an examination venture at Intel. It is at present the biggest computer vision library as far as the sheer number of capacities it holds. It is for these commending insights of OpenCV, that is utilized generally everywhere in the world.

As referenced before, OpenCV contains executions of in excess of 2500 algorithms! It is uninhibitedly accessible for business just as scholastic purposes. Also, the rundown of numerous charms of OpenCV doesn't end here! The library has interfaces for various dialects, including Python, Java, and C++. The primary OpenCV version, 1.0, was delivered in 2006 and the OpenCV people group has developed a far cry from that point forward. Presently, let us direct our concentration toward the thought behind this conversation of OpenCV, the plenty of capacities OpenCV offers.[6] We will be taking a gander at OpenCV with the point of view of Object Detection, and find out about a portion of the numerous capacities that make the undertaking of creating and understanding computer vision models simpler and charming too.[3]

5. METHODOLOGY

The software that I have developed will let you control the Portable Document Format (PDF) files and PowerPoint Presentations, without actually touching your keyboard or mouse. The main advantage of using this is that it would be easier to move around while giving presentations or scrolling the document while reading any book or article on your computer screen or watching video or listening music without touching your laptop or computer. The technologies here are OpenCV, PyAutoGUI, and NumPy and PyQt5.

The next step was to specify a lower and upper range of values for a particular color in HSV format. In the case of this software, I have specified it to yellow, since yellow is not easily found in the background. The color is specified in the RGB format. This can however be changed to a different color, according to the natural surroundings of the user. We can also set the font to suit our choice, for displaying any text on our frame. In this scenario, I have chosen the font “Hershey_Simplex”.

The below code snippet shows how to set the lower and upper range of colors in OpenCV and also how to set the font for display.

The *upper_limit* and the *lower_limit* variables are used further in the code. These values tell the frame from where it needs to create a border for tracking the movement of the object on the screen. Similarly, the *left_limit* and the *right_limit* are used to create borders for tracking the movement of the object in frame for the left and right regions, respectively.

The different actions which we need in our software are scrolling up and scrolling down, moving forward, moving backwards, and play or pause. This action could be performed by the arrow up and arrow down, left arrow, right arrow and spacebar keys

of the keyboard. Thus, I will be initializing the actions with a dictionary and default value of “false” assigned to all the tasks.

```
font = cv2.FONT_HERSHEY_SIMPLEX

yellow_lower = np.array([20, 100, 100])
yellow_upper = np.array([30, 255, 255])
upper_limit=200
lower_limit=300
left_limit=270-50
right_limit=370+25
```

Figure 2. Code snippet of specifying font and desired color

```
def pressup():
    if actions.get('up')==False:
        pyautogui.press('up')
        actions['up']=True
        print(actions)

def pressdown():
    if actions.get('down')==False:
        pyautogui.press('down')
        actions['down']=True
        print(actions)

def pressspace():
    if actions.get('space')==False:
        pyautogui.press('space')
        actions['space']=True
        print(actions)
```

Figure 3. Code snippet for action functions

The next step was to define functions for every action which needs to be performed by the user action. The actions would be *pressup, pressdown, pressleft, pressright, pressspace* and *neutral.* The *neutral* function will make all the values false for every function so that we can perform it repeatedly. The above code snippet shows the code for the functions described above. Then, I have created a while loop that would run endlessly so that our video capture does not stop abruptly even when we do not want it to stop. Inside this while loop, we would be reading our capture from the webcam. Once we start reading the capture from the webcam, it will provide us with two pieces of information. The first is the retrieval and the second one is the frame. We are interested in the frame portion because that is what would be used further in the process. This frame portion would then be utilized by the next stage in the process to evaluate the actions that need to be performed repeatedly.

```
86      def viewCam(self):
87          # read image in BGR format
88          ret, frame = self.cap.read()
89          # convert image to RGB format and put lines to divide regions
90
91          frame=cv2.cvtColor(frame,cv2.COLOR_BGR2HSV)
92          mask=cv2.inRange(frame,yellow_lower,yellow_upper)
93          contours,hierarchy=cv2.findContours(mask,cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_SIMPLE)
94          frame = cv2.line(frame,(0,198),(640,198),(255,255,255),4)
95          frame = cv2.line(frame,(0,298+45),(640,298+45),(255,255,255),4)
96          frame = cv2.line(frame,(218,0),(218,480),(255,255,255),4)
97          frame = cv2.line(frame,(368+50,0),(368+50,480),(255,255,255),4)
98          cv2.putText(frame,'Space',(50, 50),font, 1,(0, 255, 255),2, cv2.LINE_4)
99          cv2.putText(frame,'Up',(250, 50),font, 1,(0, 255, 255),2, cv2.LINE_4)
100         cv2.putText(frame,'right',(50, 250),font, 1,(0, 255, 255),2, cv2.LINE_4)
101         cv2.putText(frame,'down',(250, 400),font, 1,(0, 255, 255),2, cv2.LINE_4)
102         cv2.putText(frame,'left',(450, 250),font, 1,(0, 255, 255),2, cv2.LINE_4)
```

Figure 4. Code snippet for capturing the frame

Now since we have started reading the frame, in the next step we would want to convert this particular read frame into HSV color format to get more precise results. After converting the color format, we need to mask it so that the computer only sees the color we want it to see. This will help in making the detection more accurate as the other colors will not be visible and our color would be very easily distinguishable. This is exactly what we desire for the smooth and flawless functioning of the software.

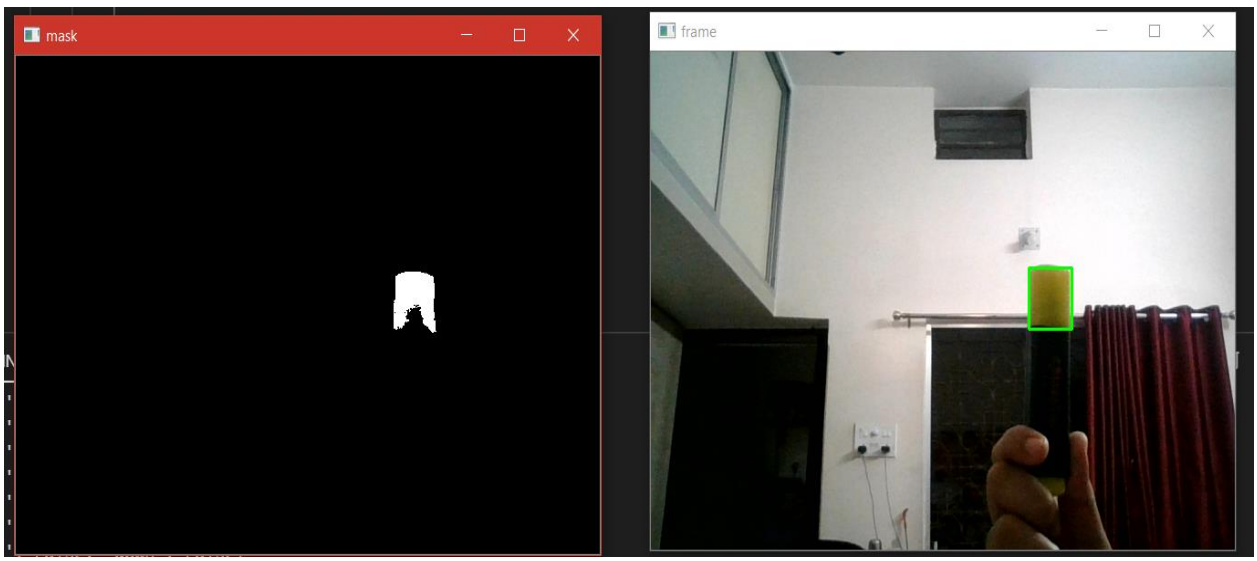

Figure 5. Frame snapshot after masking

The *hsv* variable would store the output of the *cvtColor* function, which is taking the frame and color conversion specification as input. Thus, the *hsv* variable would have our current frame in the HSV format.

The mask variable would store the result of masking the frame in the HSV format, using the i*nRange* function which specifies the yellow color's upper and lower bounds. The *cv2.line* function is an OpenCV function which is used to create visual lines for the user to know the region boundaries for different actions.

The *cv2.putText* function is used to put text on the screen for different boundaries so that the user can identify the boundaries on the screen. The above image shows the result of the frame after masking. It can be noticed from the image above that every color except the neon yellow color is masked out or ignored. We can see from the figure above that the outline of the

object is highly defined and there is no other object that can be identified. This helps us to distinguish the object of our choice from the image background to get accurate results.

Now, after identifying the object in our frame or video feed from the webcam, we need to get rid of the noise present in the image. We will be using contours from the OpenCV library to achieve the aforementioned result. Contours are lines that join continuous points in an image, which have the same color or intensity. Contours help in identifying objects and shapes as well as image recognition.

Hence, to enable contours in this software I will be using the function *cv2.findContours*. This function provides us with two outputs which are 'hierarchy' and 'contours'. The arguments which would be passed into this function would be mask since it only works on the binary image, the border definition, in this case, is "cv2.RETR_EXTERNAL". *cv2.RETR_EXTERNAL* would draw the contours as an external border of the object. The last argument to be passed is the contour approximation method. For this software, I will be using *cv2.CHAIN_APPROX_SIMPLE* as I do not want all the points across the object, rather I only want the two endpoints connecting the contours, thereby giving us a straight line and reducing the memory usage. This step has eradicated most of the noise in the image, and so we can move on to the next step. After we have drawn the contours on our object, it is time to divide the screen into different regions so that we can distinguish different object movements that the user will make to control the application

Figure 6. Frame snapshot showing action regions

This function will draw a line of the specified dimensions and in a color of our choice, to demarcate the "region of interest" in the frame. The arguments which this function takes as input are: "frame"," x coordinate", "y coordinate", "color in RGB format" and finally the "thickness of the line".

After the line has been drawn on the frame, we want to let the user know which region is used for which particular action. This will be done through the OpenCV function *cv2.putText()*. This function takes arguments "frame", "Text to be displayed", "region", "font". After doing all the above-mentioned tasks the output screen will look like this:

As it can be seen that there are 9 regions on the screen. Every region has its own functionality which will let the user to any specific function. The text in each region is written so that the user has easy navigation, and this makes the software user-friendly. Had there not been the text in the required regions, the user would have been lost and he or she would not have been

able to accomplish their goal. Now that we have detected our object and drawn contours to identify it, it is time to reduce the noise in the frame so that the computer can focus on one particular object in our webcam video feed.
To do the following task we will use the OpenCV function known as: *cv2.contourArea()*

This function calculates the area of the different contours in our video frame. I would be passing all the contours which are being read by the video feed into a "for loop". For this software, we need an area of more than 300px, this will be enough to do the work. If the contour area is greater than 300px, only then the object would be identified, otherwise, it would be neglected by the system. After we have detected the object on the screen, it is time to draw a rectangle so that we can track it in our frame. I will draw a rectangle around the object so that we can clearly distinguish it and track it on our screen.

Coming to the final step, I had to implement the actions' part. This will tell the system what action needs to be performed and when it needs to be performed, that is, the conditions in which the specific action needs to be performed. The coordinates of the rectangles drawn on the frame would come in handy for this particular task. We need its y-coordinate to know the region of the object detection in our frame, also known as the "region of interest". To do this, first, we need to define the actions that we want to perform using object movements. The different actions which we need are scrolling up and scrolling down, play/pause, move forward and move backwards.. This action could be performed by the arrow up and arrow down, spacebar, left arrow and right arrow keys of the keyboard. Thus, I will be initializing the actions with a dictionary and default value of "false" assigned to all the tasks.

Now, to perform the action, I will be using if-else conditions to distinguish between the actions. The above code snippet shows the different actions being performed for different movements of the object in the frame. The code snippet specifically shows actions for the up key, down key, left arrow key, right arrow key, spacebar, and no keypress by the keyboard.

If the object is in the upper left region, the space bar key will be pressed, similarly, if the object is in the upper middle region then the up arrow key will be pressed, if the object is in the middle left region then the left arrow key will be pressed, if the object is in the middle region then no key will be pressed, if the object is in the bottom middle region then the down arrow key will be pressed, and if the object is in the default region, that is, in the middle, then no actions will be performed and the actions would be reset. Lastly, we want to close the application as per our convenience and we do not want it to run endlessly, so we can press "q' on the keyboard to close the frame window, and all the frames opened by the program.

```
103         for c in contours:
104             # So that unneccessary objects are not detected
105             area=cv2.contourArea(c)
106             if area>300:
107                 x,y,w,h=cv2.boundingRect(c)
108                 cv2.rectangle(frame,(x,y),(x+w,y+h),(255,255,255),2)
109 
110                 if y<upper_limit and x<left_limit:
111                     pressspace()
112                 elif y<upper_limit and x>left_limit and x<right_limit:
113                     pressup()
114                 elif y>lower_limit and x>left_limit and x<right_limit:
115                     pressdown()
116                 elif y>upper_limit and y<lower_limit and x<left_limit:
117                     pressright()
118                 elif y>upper_limit and y<lower_limit and x>right_limit:
119                     pressleft()
120                 elif y>upper_limit and y<lower_limit and x>left_limit and x<right_limit:
121                     Neutral()
```

Figure 7. Code snippet showing the if-else blocks for different actions

The *Ord* function returns the ASCII value of the letter "q" and the computer checks for the keypress every 10 milliseconds. If the "q" key is pressed, it calls the break function and the loop is terminated. The function *cap.release()* tells the python interpreter that all the feed which is being read by the webcam needs to be released, and *cv2.destroyAllWindows()* function closes all the frames that were opened by the program.

Conclusion

This research work would be exceptionally helpful in corporate associations, where individuals lead enormous gatherings and introductions. The moderator would have the ability to control their introduction from a good way, without the utilization of any outside gadget like a portable remote. The individual will likewise not be needed to be physically looking through his introduction which would save time and energy, and would cause an increase in his or her mobility.

This can also be used on any web browser, and can perform the abovementioned actions of scrolling in all four directions, and playing and pausing media. For example, when visiting a website having a blog or a long article, it can be navigated without even touching the laptop, or any monitor. Only hand gestures in front of the camera can aid in easy navigation, without any hassle. Similarly, when viewing videos on online video streaming websites like YouTube, or any other OTT (Over-the-top) platforms, hand gestures can be used to play and pause the media without even bending over to your keyboard or mouse, a simple hand gesture would suffice. This is particularly useful in today's time of social distancing, when most of the population of the world is working from home and spending most of their free time binge-watching online shows and films.

The aim of this paper was to demonstrate an elaborate implementation of an Object Detection application. The software described in this report will allow you to control the Portable Document Format (PDF) records and PowerPoint Presentations, without really contacting your console or mouse. The principal benefit of utilizing this is that it is simpler to move around while giving introductions or looking over the report while perusing any book or article on your computer screen or watching video or listening to music without coming in contact with your laptop, desktop computer or even tablet. This functionality would be highly useful in corporate organizations, where people conduct large meetings and presentations. The presenter would have the power to control his or her presentation from a distance, without the use of any external device like a remote. He or she will also not be required to be manually scrolling through his presentation which would save time and energy. Likewise, when seeing videos on OTT (Over-the-top) websites or applications or online media-streaming websites like YouTube, hand motions can be utilized to play and delay the media without twisting around to your console or mouse, a basic hand motion would do the trick. This is especially valuable in the present scenario of social distancing. Since the pandemic of COVID-19 began, the greater part of the number of inhabitants on the planet is telecommuting and investing a large portion of their leisure time, watching web shows and movies.

Acknowledgements

The satisfaction that accompanies the successful completion of any task will be incomplete without the mention of the people whose ceaseless co-operation made it possible, whose constant guidance and encouragement crowns all the efforts with success.

We would also like to give our special thanks to Dr. Anil Kumar, Director, ASET who monitored our progress and gave his valuable suggestions, we choose this moment to acknowledge his contribution greatly.

I would like to express my most sincere and profound gratitude to **Dr. Sheenu Rizvi, Assistant Professor, Dept. of Computer Science and Engineering, Amity School of Engineering and Technology, Amity University Lucknow,** for his guidance throughout the work with his help and suggestions. I am also grateful to him for cooperation in providing me with all required resources. I extend special thanks to my friends and family for their constant support.

References

[1]. M. Aamir, M. Irfan, T. Ali. et al., "An adoptive threshold-based multi-level deep convolutional neural network for glaucoma eye disease detection and classification," Diagnostics (Basel), vol. 10, no. 8, Aug 2020.

[2]. N. N. Shende and S. P. Syed Ibrahim, "Layout detection using computer vision," Int. j. comput. complex. intell. algorithms, vol. 1, no. 2,

pp. 165-177, Nov 2019.

[3]. J. Bertling, “Exercising the ecological imagination: Representing the future of place,” Art educ., vol. 66, no. 1, pp. 33–39, 2013.

[4]. T. Kawasaki, “Theory of chromatography of macromolecules with rigid structures on hydroxyapatite columns. II. Dynamic part,” Biopolymers, vol. 9, no. 3, pp. 291–306, March 1970.

[5]. N. M. Jones, G. S. Paschos, B. Shrader, et al., “An overlay architecture for throughput optimal multipath routing,” in Proceedings of the 15th ACM international symposium on Mobile ad hoc networking and computing - MobiHoc ’14, pp. 73-82, Aug 2014.

[6]. A. Anderson, “A framework for NPD management: doing the right things, doing them right, and measuring the results,” Trends Food Sci. Technol., vol. 19, no. 11, pp. 553–561, Nov 2008.

[7]. S. McInnis and A. Agrawal, “Web-based visualization and navigation of the content of SNOMED CT,” in International Conference on Advances in Computing and Communication Engineering (ICACCE), June 2018.

[8]. M. N. Lambani and Z. Nengome, “Group work impact on academic communication: Female English student teachers’ views,” Int. J. Educ. Sci., vol.18, no.1–3, pp.101–109, Sep 2017.

[9]. C. P. Papageorgiou, M. Oren, and T. Poggio, “A general framework for object detection,” in Sixth International Conference on Computer Vision (IEEE Cat. No.98CH36271), pp. 555-562, Aug 2002.

[10]. G. Xie and W. Lu, “Image edge detection based on opencv,” Int. j. electron. electr. eng., vol. 1, no. 2, pp. 104–106, June 2013.

[11]. M. Rupali and P. Amit, “A review paper on general concepts of ‘artificial intelligence and machine learning,’” Int. adv. res. j. sci. eng. technol., vol. 4, no. 4, pp. 79–82, Jan 2017.

[12]. V. Wiley and T. Lucas, “Computer vision and image processing: A paper review,” Int. J. Artif. Intell. Res., vol. 2, no. 1, pp. 29-36, 2018.

[13]. H. Adusumalli, D. Kalyani, R. K. Sri, et al., “Face mask detection using OpenCV,” in Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV), 1304-1309, Feb 2021.