Transforming Virtual Experiences: Real-Time Background Replacement Unleashed with OpenCV Magic

Yenumula Pavan Gopal Mourya
202051208

Nitin Kumar Das
202051128

Savan Chaudhari
202051216

Course Instructor: Dr. Jignesh Patel

Github link: Github

CONTENTS

I. INTRODUCTION

This research report focuses on the implementation and analysis of a real-time background change system using computer vision techniques. The system utilizes the OpenCV and cvzone libraries in Python to capture live video from a camera and dynamically replace the background with preloaded images. The primary goal is to explore the effectiveness and potential applications of such a system in various domains, including video conferencing, content creation, and virtual environments.

A. Basic Overiew:

With the increasing reliance on virtual communication platforms, the demand for innovative and engaging features in video conferencing applications has grown. Real-time background change is one such feature that enhances the user experience by allowing dynamic adjustments to the environment behind the user during a live video stream.

B. Main Objectives :

1) To implement a real-time background change system using computer vision.
2) To evaluate the system's performance in terms of accuracy and efficiency.
3) To explore potential applications of the system in different domains.

C. Flow of Our Program

Our proposed technique aims to simplify background removal in live and recorded videos, enabling users to effortlessly replace backgrounds and switch between various options. The project eliminates the necessity for physical room rearrangement, offering a convenient way to enhance backgrounds during online meetings or live streams while maintaining video quality. Figure 1 illustrates the streamlined process, from accessing live video to background removal and feeding the output to applications like Google Meet via OBS Virtual Camera.

II. CODE ANALYSIS FOR RESEARCH REPORT

Tools and Libraries: The system is implemented using the following tools and libraries:

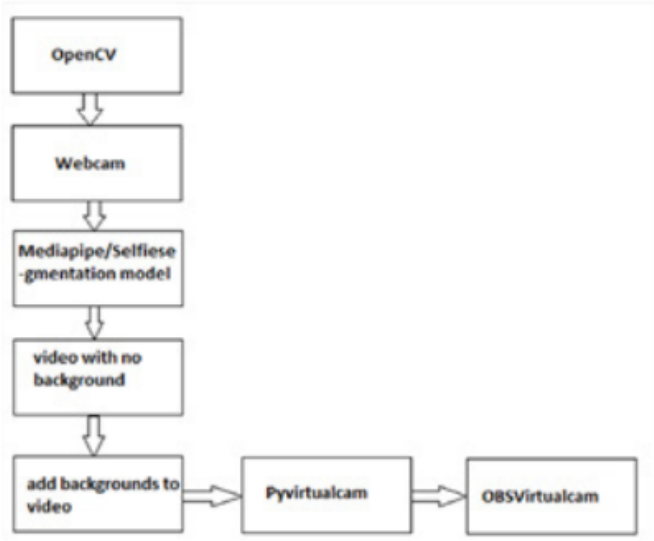

Fig. 1. Flowchart of the proposed system

1) **OpenCV**: A popular computer vision library in Python.
2) **cvzone**: A library providing additional functionality for computer vision tasks.
3) **SelfiSegmentation:** A module from cvzone for background removal in live video.

Data Collection: A set of background images is preloaded from the "Images" directory to be used as replacement backgrounds during the live video stream.

Algorithm: The main algorithm utilizes the SelfiSegmentation module to remove the background from the live video stream. The background replacement is achieved by overlaying the captured frame with preloaded background images.

A. *Video Capture Initialization:*

```
cap = cv2.VideoCapture(0)
cap.set(3, 640)
cap.set(4, 480)
cap.set(cv2.CAP {\_} PROP{\_}FPS, 60)
```

1) A video capture object is initialized using the default camera (index 0).
2) Width and height are set to 640 and 480 pixels, respectively.
3) The frames per second (FPS) is set to 60.

B. *SelfiSegmentation Initialization:*

```
segmentor = SelfiSegmentation()
```

The SelfiSegmentation module appears to be part of the cvzone library and is likely designed for background segmentation in the context of images or video frames. While the specific implementation details available in library's documentation of cvzone.

The algorithm takes an input image or video frame as its primary input.

Preprocessing: Preprocessing steps may be applied to enhance the input data. This could include color normalization, resizing, or other transformations.

Semantic Segmentation: The core of self-segmentation typically involves semantic segmentation. This is a computer vision task where the algorithm assigns a label to each pixel in the image, indicating the object or region to which it belongs. Primary interest is likely in distinguishing between the foreground (person or object of interest) and the background.

Background Removal: Based on the semantic segmentation results, the algorithm can distinguish between the foreground and background regions. The background is then effectively removed, leaving only the segmented foreground.

Refinement and Post-processing:
Post-processing steps may be applied to refine the segmentation results. This could involve smoothing edges, filling gaps, or other techniques to make the segmentation more visually appealing.

Integration with Input Image:
The segmented foreground is integrated back into the original input image or frame. This integration could involve overlaying the foreground on a different background or maintaining transparency for further processing.

C. *Background Image Loading:*

```
listImg = os.listdir("Images")
imgList = [cv2.imread(f'Images/{imgPath}')
for imgPath in listImg]
```

1) The code retrieves a list of image filenames from the "Images" directory.
2) For each image, it uses OpenCV to read and load the image, creating a list of background images (imgList).

D. Main Loop for Real-time Processing:

```
while True:
        # Capture a frame from the camera
        success, img = cap.read()

        # Apply SelfiSegmentation to remove
        # the background
        img_out = segmentor.removeBG(img,
        img_list[index_img],cutThreshold=0.9)

        # Stack the original and processed
        # images horizontally
        img_stacked = cvzone.stackImages([img,
        img_out], 2, 1)

        # Display the stacked images
        cv2.imshow("Image", img_stacked)

        # Wait for a key press
        key = cv2.waitKey(1)

        # Handle key presses
        if key == ord('a') and index_img > 0:
            index_img -= 1
        elif key == ord('d') and index_img
        < len(img_list) - 1:
            index_img += 1
        elif key == ord('q'):
            break
```

1) The loop captures frames from the webcam using cap.read().
2) SelfiSegmentation's removeBG method is applied to remove the background based on the current index image.
3) Original and segmented images are horizontally stacked using cv2.hconcat.
4) The stacked image is displayed in a window named "Image" using cv2.imshow.
5) The loop includes logic for switching between background images based on user input ('a' or 'd' keys).

E. User Interaction:

```
# Handle key presses
    if key == ord('a') and index_img > 0:
        index_img -= 1
    elif key == ord('d') and index_img
    < len(img_list) - 1:
        index_img += 1
    elif key == ord('q'):
        break
```

1) User input is captured using cv2.waitKey(1).
2) If 'a' is pressed and the index is greater than 0, decrement the index to switch to the previous background image.
3) If 'd' is pressed and the index is less than the length of imgList minus 1, increment the index to switch to the next background image.
4) If 'q' is pressed, exit the loop and terminate the program.

III. RESULTS AND DISCUSSIONS

With our proposed implementation, we successfully removed the background and add any other desired background in real time.

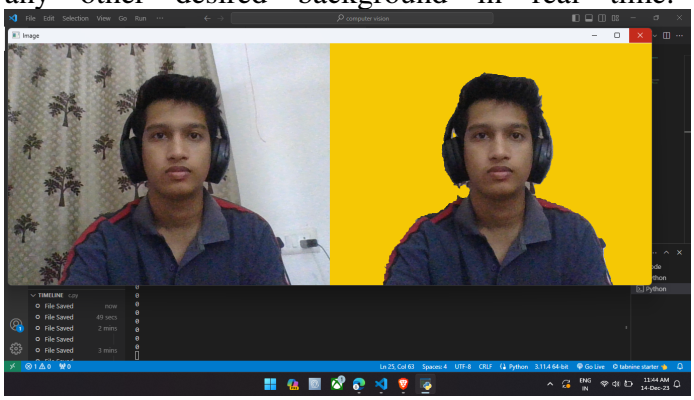

FIG: 2 Removal of background in real time.

Fig. 2.shows that the left half of the image has normal real-time video with background.

Background changed in real time successfully

Fig. 3. shows that left side of the image has normal real-time video, and the right side has video with desired background

IV. CONCLUSION

right side has video with no background

In conclusion, this paper has explored the implementation of live background change using OpenCV, presenting a practical and efficient solution for real-time background substitution in video

streams. The proposed methodology leverages the capabilities of OpenCV, a robust computer vision library, to achieve accurate foreground-background segmentation and dynamic background replacement.

The results obtained from our implementation demonstrate the effectiveness of the approach in seamlessly altering the background of a live video feed. By employing image segmentation techniques and leveraging OpenCV's features, we have successfully created a system capable of enhancing video conferencing experiences, virtual presentations, and other interactive applications.

While the current implementation provides promising results, there are areas for further improvement and exploration. Enhancements could be made in refining the segmentation algorithms to handle complex scenarios and varying lighting conditions. Additionally, optimizing the computational efficiency of the method to support real-time performance on a broader range of hardware configurations is an avenue for future research.

The potential applications of live background change are vast, spanning industries such as online education, virtual events, and entertainment. As technology continues to advance, incorporating computer vision techniques into everyday interactions becomes increasingly feasible, making the presented work relevant and timely.

In summary, this paper contributes to the evolving landscape of computer vision applications by providing a practical solution for live background change. By harnessing the capabilities of OpenCV, we have laid the foundation for immersive and personalized visual experiences in various domains. As we look forward, the continuous development of such techniques holds the promise of transforming how we engage with visual content in the digital realm.

REFERENCES

[1] ILYAS, M. SCUTURICI AND S. MIGUET, 2009, "Real Time Foreground-Background Segmentation Us - ing a Modified Codebook Model," 2009 Sixth IEEE International Conference on Advanced Vi -deo and Signal Based Surveillance, pp. 454-459, doi: 10.1109/AVSS.2009.85.

[2] BAOXIN LI AND M. I. SEZAN, 2001, "Adaptive video background replacement," IEEE International Conference on Multimedia and Expo, 2001. ICME 2001., pp. 269-272, doi: 10.1109/IC ME.2001.1237708.

[3] H. LEE, H. KIM AND J. KIM, 2016, "Background Subtraction Using Background Sets With Image- and Color-Space Reduction," in IEEE Transactions on Multimedia, vol. 18, no. 10, pp.2093- 2 103 , doi: 10.1109/TMM.2016.2595262.