INVISIBILITY CLOAK

Design Project Report

Submitted in partial fulfillment of

the requirements for the award of degree of

Bachelor Of Technology

by

SARATH SAJAN	**Reg.No : SCM17EC028**
ARDRA ANILKUMAR	**Reg.No : SCM17EC010**
AMAL T	**Reg.No : SCM17EC004**

SCMS School Of Engineering And Technology

(Affliated to A.P.J. Abdul Kalam Technological University)

Vidya Nagar, Palissery, Karukutty

Ernakulam – 683 528

November 2019

SCMS School Of Engineering And Technology

(Affliated to A.P.J. Abdul Kalam Technological University)

Vidya Nagar, Palissery, Karukutty

Ernakulam – 683 528

November 2019

BONA-FIDE CERTIFICATE

This is to certify that the design project on the topic

‘INVISIBILITY CLOAK’

by

SARATH SAJAN	**Reg.No : SCM17EC028**
ARDRA ANILKUMAR	**Reg.No : SCM17EC010**
AMAL T	**Reg.No : SCM17EC004**

Submitted in partial fulfillment of the requirement for the degree of Bachelor of Technology, is a bona-fide work carried under supervision, during the academic year 2019-2020.

Mr. Jerry Kuriakose

Ms. Anandhi

(Project Coordinators)

Dr. Saira Joseph

(Head of Department)

Table Of Contents

Chapter 1

Introduction

1.1 Abstract

Invisibility has always been a top notch super power we have always wanted and our childhood filled with magical carpets and cloaks that can make a person disappear all make it even more fascinating.

Well it may not be easy or even possible for us to make a real invisibility cloak that can bend laws of physics but we do have the power of image processing to make us invisible virtually.

In our project we are using the image processing capabilities of the OpenCV module in Python to make an **effect of invisibility in real time.** The principle is same as that of a green-screen chroma-keying.

1.2 Overview

Basically we are taking an empty shot of the background first with no moving elements in frame. The color of the cloth is predefined in HSV color space. Once the BG image is taken, execution of the program starts. The program is fed with continuous stream of images from the same camera in the same fixed position. Then whenever the cloth of predefined color enters the frame, it gets replaced by the background image taken earlier. All this calculations occur in real-time. In mainstream media and films this technique is used in the opposite way and called green-screen effect.

Chapter 2

System Study

2.1 Proposed System

- In this system we are using a webcam (mobile phone camera can be used too) to stream video for real-time processing of images.
- Python script with included OpenCV package is used to do the computational processing of the images to automatically segment out different parts of the images as per the pre-defined color values.
- A section of the image containing a particular color is removed and a background is filled in its place to give an effect of invisibility.

2.2 Advantages

- Easy installation
- Cheap alternative to green-screen equipments
- Can be executed in low-end PC's
- Robust under different lighting conditions

Chapter 3

Environmental Specifications

3.1 Hardware Requirements

- Camera (webcam/phone-camera)
- PC with good specs for smooth execution of the program

3.2 Software Requirements

- Python
- OpenCV package
- Text Editor

3.3 Miscellaneous Requirements

- A uniformly coloured cloth (preferably green)
- A camera stand

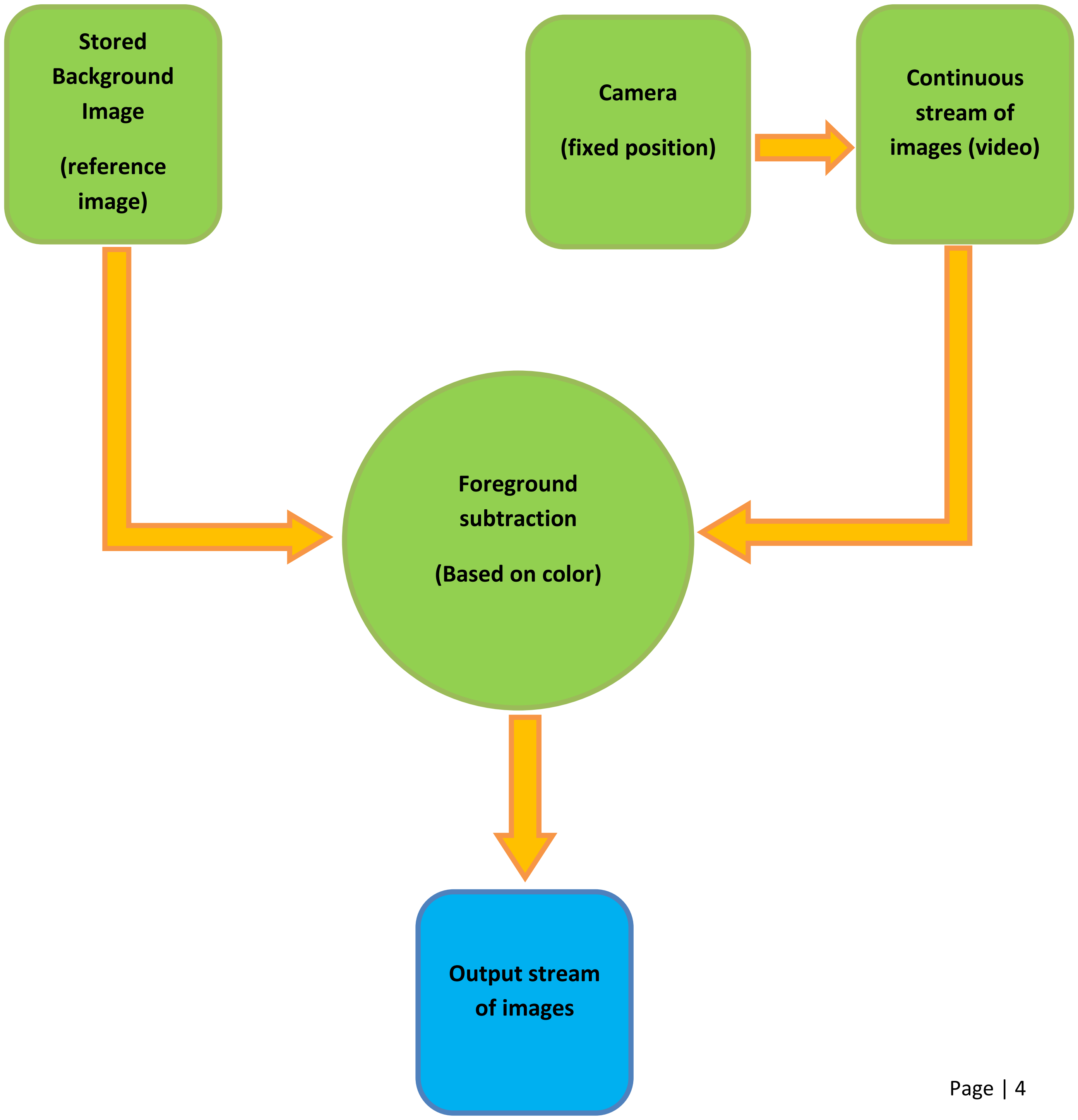
3.4 Platform and IDE

The whole system is built using the python programming language. The reason for choosing Python is because it is beginner friendly and has a huge community support and huge number of open source packages. OpenCV is one such open source computer vision package available for python for image processing techniques.

Chapter 4

Methodology

4.1 Block diagram



Stored Background Image (reference image)
Camera (fixed position)
Continuous stream of images (video)
Foreground subtraction (Based on color)
Output stream of images

The system execution can be explained in 4 stages

Stage 1 - Capture and store the background frame.

Once the program starts execution it waits for 3 seconds before capturing an image of the background with no moving objects. We call it the **empty background shot**.

This image is saved in memory and acts as the image for substitution in the main images to be captured next, which will be explained in stage 2. Once the program starts, the camera should not be moved as it will cause problems in the proper alignment of the images.

Stage 2 - Detect the red colored cloth using color detection algorithm

Now that we have our empty shot of the background, normal streaming of video starts. Subjects(humans or objects) can enter the field of view of the camera and the live feed from the camera, after processing can be seen on the monitor. For now we will fix the color for segmentation as red, so that we can use a red cloth to make the system work.

Once the program start receiving the stream of images it first converts the image from RGB color space to HSV color space. HSV color space is preferred over RGB because HSV color space is more closer to the way our eyes perceive light and color. It thus helps in identifying different shades of the same color unlike RGB. Numpy package is also used for faster computation of the image matrices. Numpy is also an Open Source package available for python for efficient numerical computations.

Stage 3 - Segment out the red colored cloth by generating a mask.

Images are captured and streamed continuously from the camera to PC. So whenever the red cloth comes in the field of view of the camera, the program detects the red color and creates a mask of black color overlaying it and thus producing a result that is foreground plus the masked black color. This main mask removes the red colored parts from current image.

Next the mask is inverted and applied on our empty background shot to remove everything from the background image except the portions in which the red color was present in the current image.

Stage 4 - Generate the final augmented output to create the magical effect.

Finally we combine the two results to get the final image where all the pixels are from the current live streamed images except for the portions with red color, which are replaced by the background empty shot image pixels and is then displayed on the screen, with all this process of capturing, processing and displaying of images taking place within 0.02 seconds.

4.2 Python Code

```
import numpy as np

import cv2

import time


time.sleep(3)


#red_low = [[0, 120, 50], [20, 255, 255]]

#red_up = [[160, 120, 50], [179, 255, 255]]


red_low = [[0, 130, 60], [15, 255, 255]]

red_up = [[164, 130, 60], [179, 255, 255]]


url = "http://192.168.1.101:8080/video"

cap = cv2.VideoCapture(url)

#time.sleep(3)

ret, background = cap.read()




while True:

    ret, img = cap.read()

    #rgb2hsv = cv2.cvtColor(img, cv2.COLOR_RGB2HSV)

    bgr2hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
    #lower red range

    lower_red = np.array(red_low[0])

    upper_red = np.array(red_low[1])

    mask1 = cv2.inRange(bgr2hsv, lower_red, upper_red)


    #upper red range

    lower_red = np.array(red_up[0])

    upper_red = np.array(red_up[1])

    mask2 = cv2.inRange(bgr2hsv, lower_red, upper_red)


    mask = mask1 + mask2


    mask1 = cv2.morphologyEx(mask, cv2.MORPH_OPEN, np.ones((3, 3),
np.uint8))

    mask1 = cv2.morphologyEx(mask1, cv2.MORPH_DILATE, np.ones((3, 3),
np.uint8))


    mask2 = cv2.bitwise_not(mask1)


    result1 = cv2.bitwise_and(img, img, mask = mask2)

    result2 = cv2.bitwise_and(background, background, mask = mask1)


    final_output = cv2.addWeighted(result1, 1, result2, 1, 0)

    cv2.imshow("background_rgb", background)
```

```
cv2.imshow('img', img)

cv2.imshow('bgr2hsv', bgr2hsv)

cv2.imshow('mask1', mask1)

cv2.imshow('mask2', mask2)

cv2.imshow('result1', result1)

cv2.imshow('result2', result2)

cv2.imshow('final', final_output)

if cv2.waitKey(25) & 0xFF == ord('q'):

    cv2.destroyAllWindows()

    break
```

Chapter 5

Conclusion and Future Scope

5.1 Conclusion

Invisibility Cloak is just one of the few, fun and creative application of image processing technique. The project also showcases how beginner-friendly python is and also the robust and powerful computer vision package OpenCV, With the help of which we were able to produce reliable and fast studio-level green screening ability. No need for specialised hardware for installing the system. It can be produced using even a mobile phone camera and a computer.

5.2 Future Scope

- Can be implemented using a more powerful PC for better frame-rate throughput.
- Ability to mask multiple colors can be added.
- The system is very flexible and can be even modified to perform motion detection or a face recognition camera for security applications.
- The whole system can built using a single Raspberry Pi and camera module thus further reducing the size considerably and improving the portability of the system

5.3 Reference

OpenCV Tutorial - https://www.learnopencv.com/invisibility-cloak-using-color-detection-and-segmentation-with-opencv/