Akshay Bankar
2019201011

Assignment-0 : Report

(One-drive link for input and output videos of following implementation : https://iiitaphyd-my.sharepoint.com/:f:/g/personal/akshay_bankar_students_iiit_ac_in/ErOJDvvZ6i5Lt_jOB60Yz9oBaFl8DhZb7Czkf0FF73qBcA?e=DoElHR)

OpenCV basic functions used:

1. Reading and writing video file:
   VideoCapture(filename)
   VideoWriter (filename, codec, fps, size, isColor)
2. Reading and writing images:
   imread(filename, any-depth/color/grey)
   imwrite(filename, input array, encoding)

Chroma keying :

Chroma keying is a technique used for combining two frames or images by replacing a color or a color range in one frame with that from the another frame. A colour range in the foreground footage is made transparent, allowing separately filmed background footage or a static image to be inserted into the scene.
Also called as matting problem— it is separation of a non-rectangular foreground image from a rectangular background image.
The principle behind chroma keying is that the color green/blue is the opposite color of skin tone, so a distinction between the two is very clear, making it easier to select the color without worrying about any part of the forgrounde object being included in the selection. The whole blue/green selection is then replaced with another frame as the background.

Algorithm used :

i. Read foreground and background videos frame-by-frame.
ii. Define a upper and lower threshold for the background color that is to be replaced.
iii. Using these threshold values, create a mask of the foreground object such that the object is balck(0) and background is white(255).
iv. Using this mask, modify the foreground RGB frame such that it only shows object and the rest of the region is black(0).
v. Similarly, modify the background frame such that the object's region is black(0).
vi. Add these foregound and backbround frames.
vii. Write the resultant frame to the Videowriter object.

OpenCV methods used :

1. VideoCapture(filename)
2. inRange(source, low_range, high_range, dest) : Checks if array elements lie between the elements of two other arrays.
Parameters:
- source – first input array.
- low_range – inclusive lower boundary array or a scalar.
- high_range – inclusive upper boundary array or a scalar.

- dest – output array of the same size as `src` and `CV_8U` type.

Face detection :
The face detection is performed using the OpenCV's haar classifier face which is a Cascade classifier based on Viola-Jones algorithm.
The classifier:
The classifier is trained using edge, lines and rectangle haar features. Once the feature is obtained, it is given weights according to its accuracy of detecting the faces correctly.
To reduce the number of features, adaboost technique is used which extracts the important features.
To build a strong classifier, several weak classifier are cascaded such that they reject a non-face fast and allows possible face regions to go through the higher stage of the cascade.
Detection:
On the input image, the haar features are calculated in specific window sizes and the value for features threshold of that stage in the classifier. If the window is above the threahold it is considered for the next stage else it is rejected. This is done on multple scales so as to detect faces with varying sizes.
Algorithm :

i. Load the haar cascade classfier for the face .
ii. Open the webcam using the VideoCapture object of opencv.
iii. Read the frame and convert it to gray.
iv. Give this gray-scale image to the multisclae detector which does face detection at multiple scales of the image pyramid.
v. The detector returns a vector containing top start position of the ROI and its width and height.
vi. Draw all such returned ROIs on the frame.

OpenCV methods used :

1. VideoCapture(filename)
2. CascadeClassifier(filename) : Loads a classifier from the file.
3. detectMultiScale(input_img, Rect_objects, scaleFactor=1.1, minNeighbors, minSize, maxSize) : Detects objects of different sizes in the input image. The detected objects are returned as a list of rectangles.

Parameters :
- cascade – Haar classifier cascade.
- image – Matrix of the type CV_8U containing an image where objects are detected.
- objects – Vector of rectangles where each rectangle contains the detected object.
- scaleFactor – Parameter specifying how much the image size is reduced at each image scale.
- minNeighbors – Parameter specifying how many neighbors each candidate rectangle should have to retain it.
- minSize – Minimum possible object size. Objects smaller than that are ignored.
- maxSize – Maximum possible object size. Objects larger than that are ignored.