#### **ABSTRACT:**

**Face detection** is a computer technology being used in a variety of applications that identifies human faces in digital images.

Face detection algorithms automatically tracks the position and rotation of the face in the picture or video, which allows for blurring out faces.

One of the applications of face blurring is for the purpose of **privacy protection** in Google Street View.

#### TECHNICAL DISCUSSION

A Haar Cascade is used in this project which is basically a classifier to detect particular objects from the source. The haarcascade\_frontalface\_default.xml is a haar cascade designed by OpenCV to detect the frontal face. A Haar Cascade works by training the cascade on thousands of negative images with the positive image superimposed on it. The Haar cascade is capable of detecting features from the source.

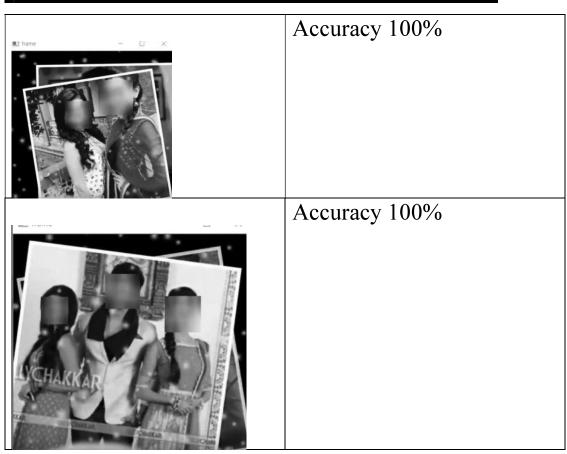
## **RESULTS:**

### First Video:

Link:

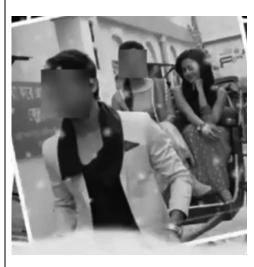
https://www.youtube.com/watch?v=4B0YBT6FQAc&feature=youtu.be

```
C:\Users\Diana Atef\Desktop\Assignment>py try2.py
Frames per second
29.152
Total Number of frames
948
```





# Accuracy 100%

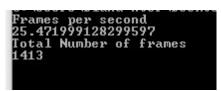


Accuracy 90%

## Second video:

### Link:

https://www.youtube.com/watch?v=qtE1xM1oOQs&featur e=youtu.be



Accuracy 100%
Accuracy 100%
Accuracy 100%

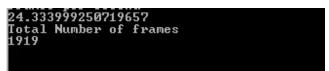


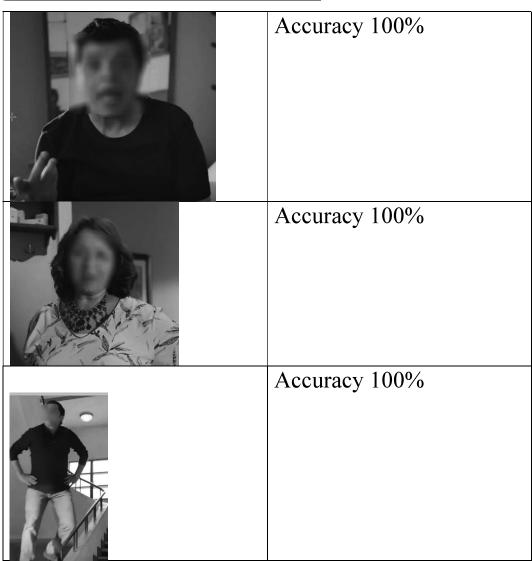
Accuracy 0%
Face not detected

### Third video:

Link:

https://www.youtube.com/watch?v=yj7jloz1lsE&feature=youtu.be





Timing in real time using number of frames=1413

Fps is less than that of an input video with the same frame number

#### FOR THE REAL TIME:

C:\Users\Diana Atef\Desktop\Assignment>py real\_time.py 16.96626966599421

#### FOR AN INPUT VIDEO:

Frames per second 25.471999128299597 Total Number of frames 1413

#### **APPENDIX**

#### CODE USED FOR AN INPUT VIDEO

```
import numpy as np
import cv2
#Haarcascade classifier is used for frontal face detection
face cascade =
cv2.CascadeClassifier('haarcascade frontalface default.xm
1')
#input video to be captured
cap = cv2.VideoCapture('3.mp4')
#save the output video
fourcc = cv2.VideoWriter fourcc(*'XVID')
out = cv2. VideoWriter('output.avi', fource, 20.0, (320,240))
#function used to get frames per second
fps = cap.get(cv2.CAP_PROP_FPS)
print ("Frames per second")
print(fps)
#function used to get total number of frames
total= int(cap.get(cv2.CAP PROP FRAME COUNT))
print ("Total Number of frames")
print(total)
```

```
detecting faces
while(cap.isOpened()):
         ret, frame = cap.read()
         faces = face cascade.detectMultiScale(frame,
1.3, 5)
         for (x,y,w,h) in faces:
              frame[y:y+h, x:x+w]
  # apply a gaussian blur on this new rectangle image
              frame[y:y+h,x:x+w] =
cv2.GaussianBlur(frame[y:y+h,x:x+w],(23, 23), 30)
              out.write(frame)
              cv2.imshow('frame',frame)
         if cv2.waitKey(1) & 0xFF == ord('q'):
              break
cap.release()
cv2.destroyAllWindows()
```

#while video has not reached its end keep reading and

```
For Real time process
import numpy as np
import cv2
face cascade =
cv2.CascadeClassifier('haarcascade frontalface default.xm
1')
#capturing video from camera
cap = cv2.VideoCapture(0)
fps = cap.get(cv2.CAP PROP FPS)
print ("Frames per second")
print(fps)
total= int(cap.get(cv2.CAP_PROP_FRAME_COUNT))
print ("Total Number of frames")
print(total)
while(cap.isOpened()):
         ret, frame = cap.read()
         faces = face_cascade.detectMultiScale(frame,
1.3, 5)
         for (x,y,w,h) in faces:
              frame[y:y+h, x:x+w]
  # apply a gaussian blur on this new recangle image
```

```
frame[y:y+h,x:x+w] =
cv2.GaussianBlur(frame[y:y+h,x:x+w],(23, 23), 30)
              cv2.imshow('frame',frame)
         if cv2.waitKey(1) & 0xFF == ord('q'):
              break
cap.release()
cv2.destroyAllWindows()
For calculating real timing
import numpy as np
import cv2
import time
face cascade =
cv2.CascadeClassifier('haarcascade_frontalface_default.xm
1')
cap = cv2.VideoCapture(0)
#specifying number of frames
num frames = 1413
start = time.time()
for i in range(0, num frames):
         ret, frame = cap.read()
```

```
faces = face cascade.detectMultiScale(frame,
1.3, 5)
         for (x,y,w,h) in faces:
              frame[y:y+h, x:x+w]
  # apply a gaussian blur on this new rectangle image
              frame[y:y+h,x:x+w] =
cv2.GaussianBlur(frame[y:y+h,x:x+w],(23, 23), 30)
              cv2.imshow('frame',frame)
         if cv2.waitKey(1) & 0xFF == ord('q'):
              break
end = time.time()
seconds = end - start
fps = num_frames / seconds
print (fps)
cap.release()
cv2.destroyAllWindows()
```