Realtime Blurring Camera

By 5th Group :
Romain Monnier
Salma Rahma Lailia
신희재

Goals

-Create a Realtime Blurring Camera

- Vloggers and Live Streamers are on the rise these days.
- When they take a video, inevitably someone else is taken.
- Therefore, we wanted to implement a function that automatically hides the faces of other people besides the registered ones.

What We Did

-Create a registration process

- Take multi-directional face photos of the person you wish to register
- Train the AI with those photos

-Create and test the blurring code

- When viewing a registered user, it does not blur and outputs the name
- Blur unregistered users

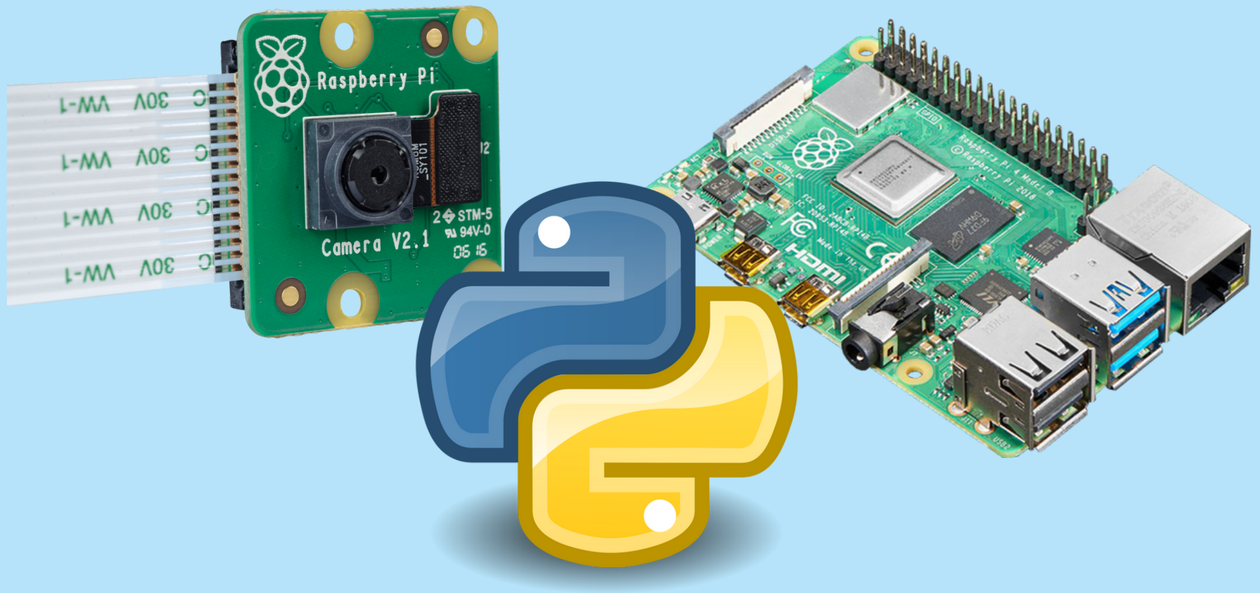
Tools

-Software:

- Python

-Hardware:

- Raspberry Pi 32bit
- Raspberry Pi Camera
- HDMI Cable
- Computer



Process

face_shot.py

↓

train_model.py

↓

blurryv3.py

face_shot.py

```python
import cv2

name = 'Heejae' #replace with your name

cam = cv2.VideoCapture(0)

cv2.namedWindow("press space to take a photo", cv2.WINDOW_NORMAL)
cv2.resizeWindow("press space to take a photo", 500, 300)

img_counter = 0

while True:
    ret, frame = cam.read()
    if not ret:
        print("failed to grab frame")
        break
    cv2.imshow("press space to take a photo", frame)

    k = cv2.waitKey(1)
    if k%256 == 27:
        # ESC pressed
        print("Escape hit, closing...")
        break
    elif k%256 == 32:
        # SPACE pressed
        img_name = "dataset/"+ name +"/image_{}.jpg".format(img_counter)
        cv2.imwrite(img_name, frame)
        print("{} written!".format(img_name))
        img_counter += 1

cam.release()

cv2.destroyAllWindows()
```

This code is photographing code. It takes a picture every time press the 'space', and then it will stores the pictures in specific directory with typed name

train_model.py

This code will find images in the 'dataset' folder, and train the model with those images

```python
#! /usr/bin/python

# import the necessary packages
import imutils
from imutils import paths
import face_recognition
#import argparse
import pickle
import cv2
import os

# our images are located in the dataset folder
print("[INFO] start processing faces...")
imagePaths = list(paths.list_images("dataset"))

# initialize the list of known encodings and known names
knownEncodings = []
knownNames = []

# loop over the image paths
for (i, imagePath) in enumerate(imagePaths):
    # extract the person name from the image path
    print("[INFO] processing image {}/{}".format(i + 1,
        len(imagePaths)))
    name = imagePath.split(os.path.sep)[-2]

    # load the input image and convert it from RGB (OpenCV ordering)
    # to dlib ordering (RGB)
    image = cv2.imread(imagePath)
    rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

    # detect the (x, y)-coordinates of the bounding boxes
    # corresponding to each face in the input image
    boxes = face_recognition.face_locations(rgb,
        model="hog")

    # compute the facial embedding for the face
    encodings = face_recognition.face_encodings(rgb, boxes)

    # loop over the encodings
    for encoding in encodings:
        # add each encoding + name to our set of known names and
        # encodings
        knownEncodings.append(encoding)
        knownNames.append(name)

# dump the facial encodings + names to disk
print("[INFO] serializing encodings...")
data = {"encodings": knownEncodings, "names": knownNames}
f = open("encodings.pickle", "wb")
f.write(pickle.dumps(data))
f.close()
```

blurryv3.py

```python
from imutils.video import VideoStream
from imutils.video import FPS

import face_recognition
import imutils
import pickle
import time
import cv2

currentname = "unknown"
encodingsP = "encodings.pickle"
cascade = "haarcascade_frontalface_default.xml"

print("[INFO] loading encodings + face detector...")
data = pickle.loads(open(encodingsP, "rb").read())
detector = cv2.CascadeClassifier(cascade)

print("[INFO] starting video stream...")
vs = VideoStream(src=0).start()

time.sleep(2.0)

fps = FPS().start()



def find_and_blur(gray, frame):
        # detect all faces
        rects = detector.detectMultiScale(gray, scaleFactor=1.1,
```

```python


def find_and_blur(gray, frame):
        # detect all faces
        rects = detector.detectMultiScale(gray, scaleFactor=1.1,
        minNeighbors=5, minSize=(30, 30),
        flags=cv2.CASCADE_SCALE_IMAGE)
        # get the locations of the faces
        for (x, y, w, h) in rects:
            # select the areas where the face was found
            roi_frame = frame[y:y+h, x:x+w]
            # blur the colored image
            blur = cv2.GaussianBlur(roi_frame, (99,99),30)
            # Insert ROI back into image
            frame[y:y+h, x:x+w] = blur
    # return the blurred image
        return frame

while True:
    frame = vs.read()
    frame = imutils.resize(frame, width=500)
    # transform color -> grayscale
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    rects = detector.detectMultiScale(gray, scaleFactor=1.1,
        minNeighbors=5, minSize=(30, 30),
```

<Blurring Section>
Blur the selected section

blurryv3.py

```
42          return frame
43
44  while True:
45      frame = vs.read()
46      frame = imutils.resize(frame, width=500)
47      # transform color -> grayscale
48      gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
49      rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
50      rects = detector.detectMultiScale(gray, scaleFactor=1.1,
51          minNeighbors=5, minSize=(30, 30),
52          flags=cv2.CASCADE_SCALE_IMAGE)
53      boxes = [(y, x + w, y + h, x) for (x, y, w, h) in rects]
54      encodings = face_recognition.face_encodings(rgb, boxes)
55      names = []
56
57      for encoding in encodings:
58          matches = face_recognition.compare_faces(data["encodings"],
59              encoding)
60          name = "Unknown"
61          if True in matches:
62              matchedIdxs = [i for (i, b) in enumerate(matches) if b]
63              counts = {}
64              for i in matchedIdxs:
65                  name = data["names"][i]
66                  counts[name] = counts.get(name, 0) + 1
67                  name = max(counts, key=counts.get)
68                  if currentname != name:
69                      currentname = name
70                      print(currentname)
71          names.append(name)
72          if name == "Unknown":
73                  blur = find_and_blur(gray, frame)
74      for ((top, right, bottom, left), name) in zip(boxes, names):
```

<Default Section>
In the while loop, it keeps taking video

```
74      for ((top, right, bottom, left), name) in zip(boxes, names):
75          # draw the predicted face name on the image - color is in BGR
76          cv2.rectangle(frame, (left, top), (right, bottom),
77              (0, 255, 0), 2)
78          y = top - 15 if top - 15 > 15 else top + 15
79          cv2.putText(frame, name, (left, y), cv2.FONT_HERSHEY_SIMPLEX,
80              .8, (255, 0, 0), 2)
81      cv2.imshow("Facial Recognition is Running", frame)
82      key = cv2.waitKey(1) & 0xFF
83
84      if key == ord("q"):
85          break
86      fps.update()
87  fps.stop()
88  print("[INFO] elasped time: {:.2f}".format(fps.elapsed()))
89  print("[INFO] approx. FPS: {:.2f}".format(fps.fps()))
90  cv2.destroyAllWindows()
91  vs.stop()
92
93
94
```

<Detecting Section>
If someone detected is unknown person, it will blur the person

<Description Section>
This part will shows up the name of the person. By the upper codes, unknown person's name is 'Unknown'

Video:

Access link for full code: https://github.com/salmalailia/RealtimeBlurryCamera.git