Crowd Motion Analysis In Deep Learning

Name: Amandeep
Section: H
Rollno: 1918199/32

Introduction

Crowds or large gatherings in various places such as entertainment events, airports, hospitals, stadiums, theme parks that deal with individuals almost daily. Activities are also very different from social and cultural religions. Unlike social and sport-related events, crowd-related situations Religious events such as Newyear and Diwali may not be possible to avoid .Therefore it is important to have a smart CrowdMonitoring System (CMS) to ensure security community , maintain high pedestrian traffic to prevent stamps , provision better emergency services in case of crowd -related emergencies and development services that provide good access by avoiding congestion . Negative ideas , crowd management , monitoring , and analytic shave potential fora number of applications.

Algorithms used in Motion Analysis

We have used OpenCV in order the build this project. The geometric approach focuses on distinguishing features. The photo-metric statistical methods are used to extract values from an image. These values are then compared to templates to eliminate variances. The algorithms can also be divided into two more general categories — feature-based and holistic models. The former focuses on facial landmarks and analyzes their spatial parameters and correlation to other features, while holistic methods view the human face as a whole unit.

HAAR Cascade Algorithm

Haar Cascade classifiers are an effective way for object detection. ... Haar Cascade is a machine learning-based approach where a lot of positive and negative images are used to train the classifier.

We apply every feature of the algorithm on all the training images. Every image is given equal weight at the starting. It founds the best threshold which will categorize the faces to positive and negative. There may be errors and misclassifications. We select the features with a minimum error rate, which means these are the features that best classifies the face and non-face images.

Simple Model In Computer Vision

- ***Training data.*** *This type of data builds up the machine learning algorithm. The data scientist feeds the algorithm input data, which corresponds to an expected output. The model evaluates the data repeatedly to learn more about the data's behavior and then adjusts itself to serve its intended purpose.*

- ***Test data.*** *After the model is built, testing data once again validates that it can make accurate predictions. If training and validation data include labels to monitor performance metrics of the model, the testing data should be unlabeled. Test data provides a final, real-world check of an unseen dataset to confirm that the ML algorithm was trained effectively.*

Motion Detection Code

```
import cv2

face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')


cap = cv2.VideoCapture(0)


while True:

    _, img = cap.read()

    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

    faces = face_cascade.detectMultiScale(gray, 1.1, 4)

    for (x, y, w, h) in faces:
        cv2.rectangle(img, (x, y), (x+w, y+h), (255, 0, 0), 2)

    cv2.imshow('img', img)

    k = cv2.waitKey(30) & 0xff
    if k==27:
        break
```

Object Detection Using Colors

```
import cv2
import numpy as np

# Webcamera no 0 is used to capture the frames
cap = cv2.VideoCapture(0)

def nothing(x):
    pass
cv2.namedWindow("Tracking")
cv2.createTrackbar('LH', 'Tracking', 0, 255, nothing) # LH(lower hue) nothing function
is just for formality.
cv2.createTrackbar('LS', 'Tracking', 0, 255, nothing) #LS(lower saturation)
cv2.createTrackbar('LV', 'Tracking', 0, 255, nothing) #LV(lower value)
cv2.createTrackbar('UH', 'Tracking', 255, 255, nothing) #UH(upper hue)
cv2.createTrackbar('US', 'Tracking', 255, 255, nothing) #US(upper saturation)
cv2.createTrackbar('UV', 'Tracking', 255, 255, nothing)
```

```
while(1):
    # Captures the live stream frame-by-frame
    _, frame = cap.read()
    # Converts images from BGR to HSV
    hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)

    l_h = cv2.getTrackbarPos('LH', "Tracking")
    l_s = cv2.getTrackbarPos('LS', "Tracking")
    l_v = cv2.getTrackbarPos('LV', "Tracking")

    u_h = cv2.getTrackbarPos('UH', "Tracking")
    u_s = cv2.getTrackbarPos('US', "Tracking")
    u_v = cv2.getTrackbarPos('UV', "Tracking")

    lower_blue = np.array([l_h, l_s, l_v])
    upper_blue = np.array([u_h, u_s, u_v])

    # Here we are defining range of bluecolor in HSV
    # This creates a mask of blue coloured
    # objects found in the frame.
    mask = cv2.inRange(hsv, lower_blue, upper_blue)

    # The bitwise and of the frame and mask is done so
    # that only the blue coloured objects are highlighted
    # and stored in res
    res = cv2.bitwise_and(frame,frame, mask= mask)
    # cv2.imshow('frame',frame)
    cv2.imshow('mask',mask)
```

Motion Detection Using OpenCV

```
import cv2
import numpy as np

cap = cv2.VideoCapture("vedio.mp4")

ret, frame1 = cap.read()
ret, frame2 = cap.read()

while(cap.isOpened()):
    diff = cv2.absdiff(frame1, frame2)  # (absolute difference)function is use
to find the diff. in 1st nd 2nd frame in video.
    gray = cv2.cvtColor(diff, cv2.COLOR_BGR2GRAY) # converting diff in
grayscale mode because we find contours in latest stages of vedio.
    # we know that gray scale is bst mode to find contours in image.
    blur = cv2.GaussianBlur(gray, (5,5), 0) # bluring the diff image.
```

```
_, thresh = cv2.threshold(blur, 20, 255, cv2.THRESH_BINARY) # appliying threshhold technique in
blur image.
    # then we dilate our image to fill all holes in our thresh image to find out better contours in
image.
    dilated = cv2.dilate(thresh, None, iterations=3) # iterations mean layer which we want to cover
holes (we now previous vedios)
    contours, _ = cv2.findContours(dilated, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)

    # we want rectangles in motion peoples not contours, so we itterate over all the contours.
    for contour in contours:
        (x, y, w, h) = cv2.boundingRect(contour) # it gives us x, y, width and height of frame.

        if cv2.contourArea(contour) <300: # means if area is very small than we don't want to do
anything.
            continue
        cv2.rectangle(frame1, (x, y), (x+w, y+h), (0, 255, 0), 2)
        cv2.putText(frame1, 'STATUS: {}'.format('Movement'), (10, 20), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0,
255), 3)
    #cv2.drawContours(frame1, contours, -1, (0,255,0), 2 ) # we going to draw contour in only in
our frame 1.

    cv2.imshow('vedio', frame1)
    frame1 = frame2
    ret, frame2 = cap.read() # this means 3rd frame is assigned into frame2 variable and so on.



    if cv2.waitKey(40) == 27:  #the argument of waitkey is (40milisecond), the function wait for 40ms
for next frame
```

Application

• Military Applications : Number of combat jets, soldiers, and moving drones as well their movements etc. measured by appropriate crowd management systems . So i military capabilities can be measured using this system.

• Security Monitoring: Large number of CCTV monitoring systems installed various places such as religious gatherings , airports , and public places allow better crowd monitoring programs .For example, an analytical system has been developed behavior and congestion times that ensure safety and security .Similarly , gifts a new way to identify risks by analyzing population congestion .Good monitoring is proposed a system that produces a visual report on the analysis of the crowd and its flow indifferent directions.

• Disaster Management : There are various congestion conditions such as music concerts and sporting events etc ., where half of the crowd charges randomly directions, which create life-threatening situations.

ThankYou