EE417 FINAL PROJECT

Mustafa Cem Büyükalpelli 29507

Content

Importance of the Problem and Problem Definition

In just year of 2021 more than a half million people went missing in just the United states of america moreover more than two thousand people are going missing in just India. I do think this is a huge problem. Unfortunately, searching for these people is both really expensive and labor intensive. I believe that if we augment this process with drones that uses computer vision we can have faster and more successful searches with less people and money which would be beneficial for everyone included. Of course Drones have the advantage of aerial vision and we can equip them with powerful cameras for these rescue operations. Moreover i believe that computer vision is the perfect tool to solve this problem because human vision is really flawed and vulnerable towards exhaustion. With a solution that will aerilly detect i believe will unify many people with their loved ones and Save money ın the process.

Problem Formulation and Solution Method

Problem Formulation: Develop an aerial human detection software system that can accurately and efficiently identify and locate individuals in real-time within aerial footage captured by drones or other aerial vehicles, in various lighting and weather conditions, and with minimal latency and high frame rates for real-time decision making. The system should also be robust to common challenges such as occlusions and variations in camera angles, and provide alerts for potential safety or security concerns.

Solution Method:

Data Collection:I gathered about 350 images with different lighting,distance, background and angels. Moreover also copies of some these images with turned, slanted and with different projections so i can have a bigger dataset.

Pre-processing: Perform image pre-processing techniques such as noise reduction, color space conversion, and histogram equalization to improve the quality of the images also for training purposes and to keep performance better all images are converted 600x800 resolution.

Feature Extraction: Extract relevant features from the images using techniques such as Haar cascades

Training:Training is done YOLO V8 originally it was trained with YOLO V7 but since project got a extension and YOLO V8 recently came out(ı had problem with yolo v8’s predict function so i made very small modifications), model is retrained project is adopted to that. Also training done with 200 epoch (Bhalerao 2023)

Model Evaluation: Model has accuracy of 84 percent and unfortunately it has low tendency provide false positive when pictures with thick poles as wide as humans are given this is probably because vast majority of images given are upright.

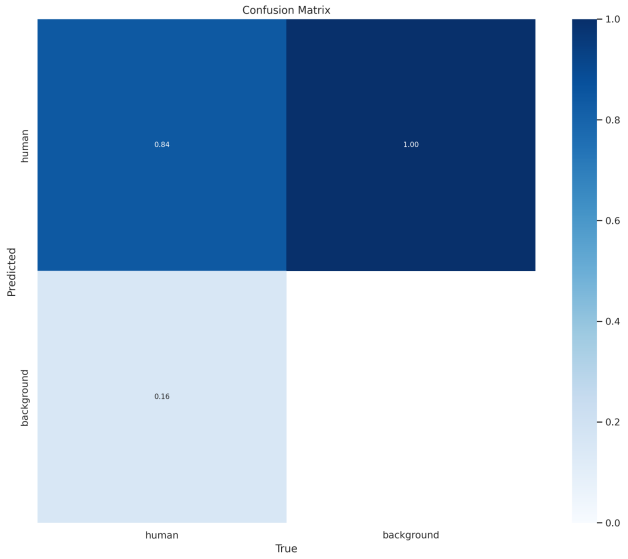
Real-time Implementation: Software is takes 5 photos sequentially choose the least blurry and perform the object detection.

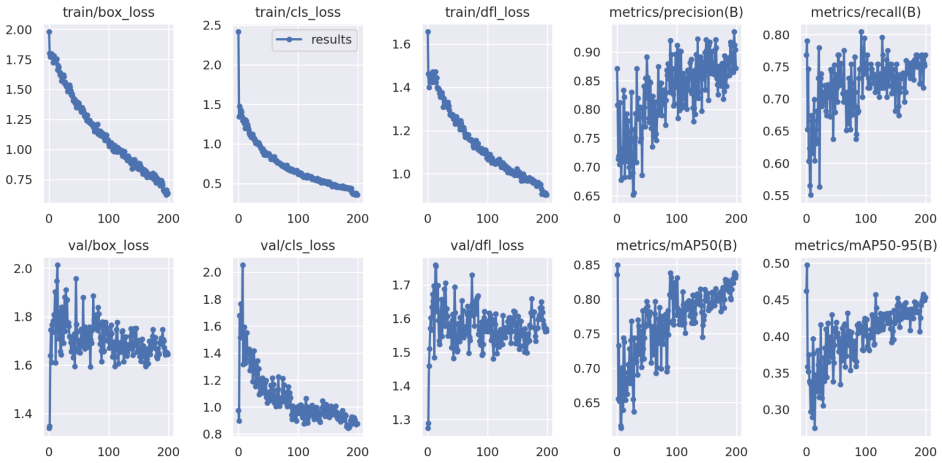
Implementation and results

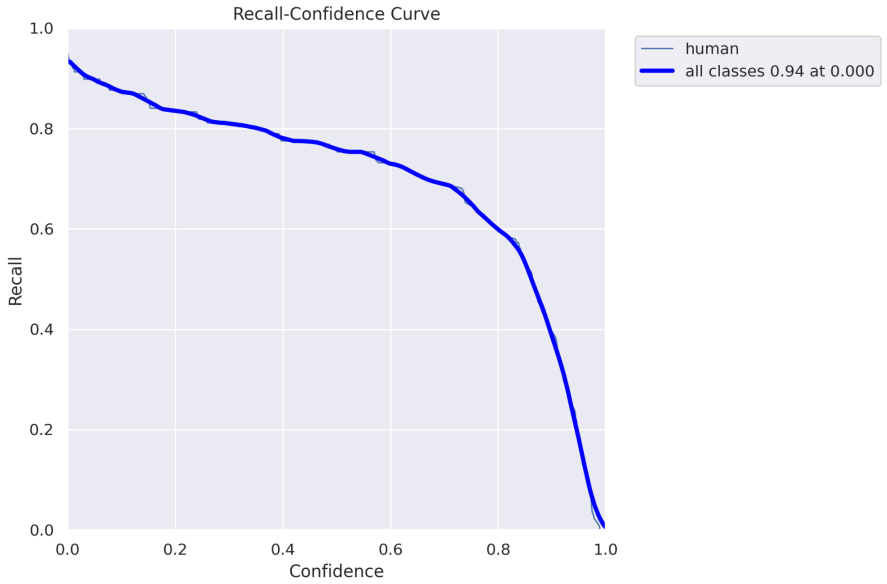
İmplementation:

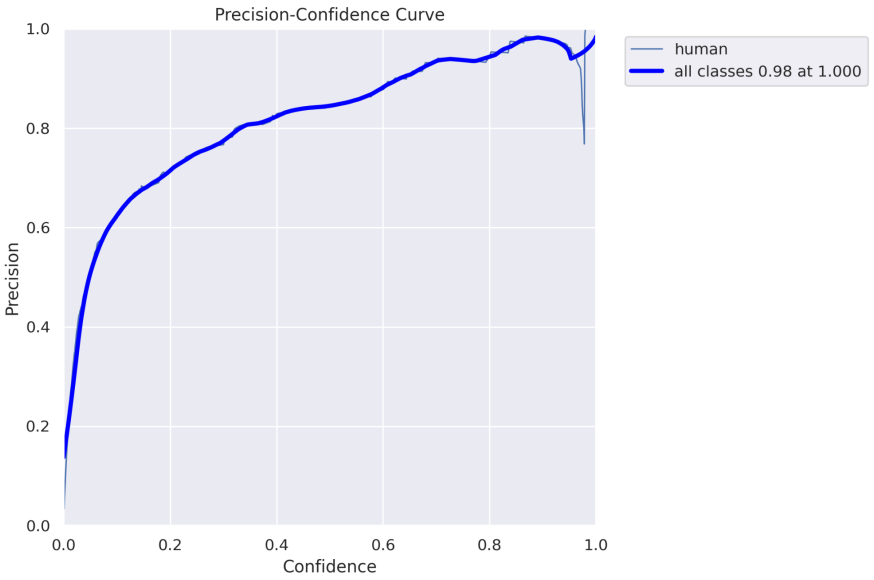
Because this software will run drone and drones as vulnerable to wind and other external elements I decided instead of feeding live video to ai to sequentially take 5 photos and choose the less blurry image by comparing laplacian values. This approach also requires less computational power if processing done externally one computer can power more drones this way ıf process is done on a drone ıt will consume less electricity and make it cheaper. Of course it wouldn't be as smooth as direct video. But since the purpose is to detect people in a wide area this approach is fitting and more logical. Also we can choose to apply linear filter if camera is low quality and add sharpening to image for rainy weathers and finally ai object detection is performed.

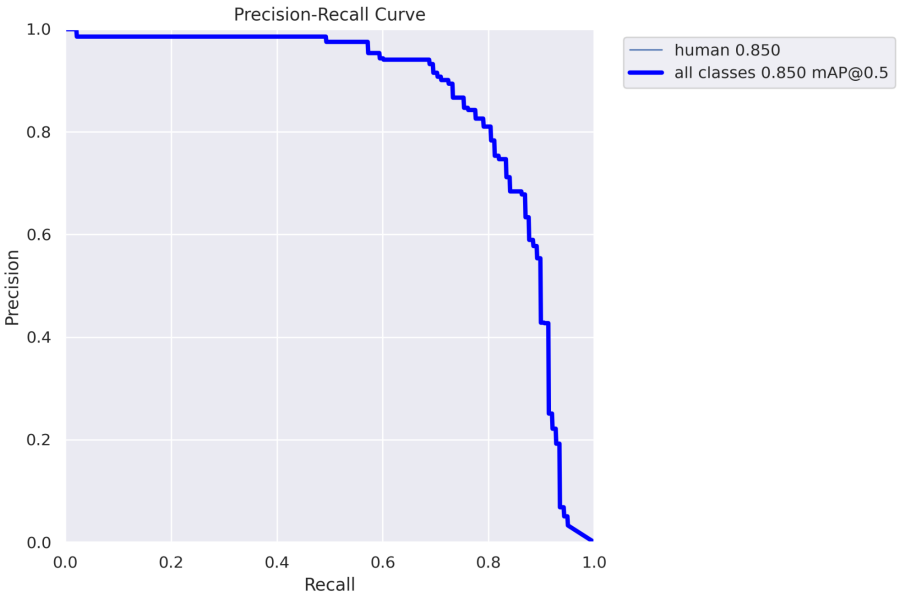
results:

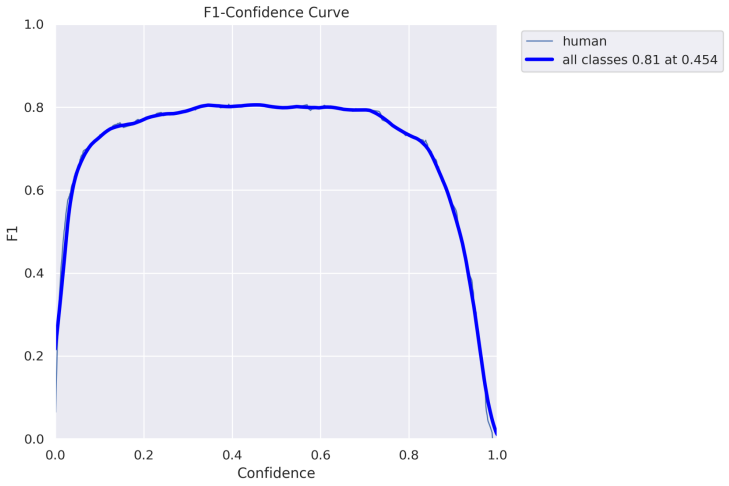



train/box_loss
train/cls_loss
results
train/dfl_loss
metrics/precision(B)
metrics/recall(B)
val/box_loss
val/cls_loss
val/dfl_loss
metrics/mAP50(B)
metrics/mAP50-95(B)


Recall-Confidence Curve
human
all classes 0.94 at 0.000
Recall
Confidence


Precision-Confidence Curve
human
all classes 0.98 at 1.000
Precision
Confidence


Precision-Recall Curve
human 0.850
all classes 0.850 mAP@0.5
Precision
Recall
0.0
0.2
0.4
0.6
0.8
1.0


F1-Confidence Curve
human
all classes 0.81 at 0.454
F1
Confidence
0.0
0.2
0.4
0.6
0.8
1.0

Sharpened image:

blurry image:

Discussion of the results

I do genuinely think that if developed more this can be a great solution for finding missing people. There is still a lot of room for improvement. The first thing I would do to improve the system is add thick poles as a second class of object detection to avoid false positives and I would also increase the amount of images. Moreover i would add more filters and image process to against more defects and weather conditions. But I do think choosing to just 1 frame out 5 is a good and sound application for this case and implementation is good enough to be included in another more advanced version of this software. I think this results are satisfactory or its purpose and it is in useful condition. There was also optical flow feature but it is removed because it doesn’t provide good enough and slowed down the processes considerably . I think there is still a lot of room for improvement and i would like to improve it more.

Appendix

```
import cv2
import time
import numpy as np
from ultralytics import YOLO

model = YOLO("best.pt")
def sarphen(image):
    kernel = np.array([[0, -1, 0],
                       [-1, 5, -1],
                       [0, -1, 0]])
    sharp = cv2.filter2D(src=image, ddepth=-1, kernel=kernel)
    return sharp
def lin_blur(image):
    sharp = cv2.medianBlur(image,3)
    return sharp


def blur_finder():
    val1 = 0
    pos = 0
    for x in range(5):
        img =
cv2.imread('C:/Users/masdaq/Desktop/ocv/ultralytics-main/opencv_
frame_' + str(x) + '.png')
        gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
        val0 = cv2.Laplacian(gray, cv2.CV_64F).var()
        print(val0)
        if val0 >= val1:
            val1 = val0
            pos = x
            if (an0 == "y" or an1 == "y"):
```

```
            if (an0 == "y" and an1 == "y"):
                img = sarphen(img)
                img = lin_blur(img)
            elif(an0 == "y" and an1 == "n"):
                img = sarphen(img)
            else:
                img = lin_blur(img)
            img = cv2.imwrite('C:/Users/masdaq/Desktop/ocv/ultralytics-main/opencv_frame_' + str(pos) + '.png', img)



    predictions = model.predict(source='opencv_frame_' + str(pos) + '.png', show=wns, conf=0.5, device = dvc )
    #model.predict(source=0, show=wns, conf=0.5, device=dvc)#too see live video
    inf0 = ""
    inf1 = ""
    inf2 = ""
    temp0 = 0
    temp1 = 0
    for x in range(len(predictions)):
        print("[T]")
        # print(predictions[x])
        if x == 0:
            # temp1 = predictions[x].find("[")
            inf0 = predictions[x]
            inf0 = str(inf0).split()
            temp1 = inf0.index("shape:")
            inf0 = str(inf0[temp1 + 1:temp1 + 3])
            inf0 = [int(i) for i in inf0 if i.isdigit()]
            print("************************************")
            print("!!Number of people:    "+str(inf0[0]))
            print("************************************")
```

```
        elif x == 1:
            inf1 = predictions[x]
    print(str(pos) +" is the most clear one with "+str(val1))

an0 =  ""
while (an0 == "y" or an0 == "n") == False :
    an0 = input("Would you like to apply sharpening? we dont usally
recommond this on normal weathers it is recommended for rainy
and foggy weathers y/n ")
an1 =  ""
while (an1 == "y" or an1 == "n") == False :
    an1 = input("Would you like to apply linearfilter? against noise the
cheaper your camera more you would like to aplly this y/n ")
an2 =  ""
while (an2 == "y" or an2 == "n") == False :
    an2 = input("Does your pc supporrt CUDA and is it set up? y/n ")
an3 =  ""
while (an3 == "y" or an3 == "n") == False :
    an3 = input("would you like to see images with detection? y/n ")
if an2 == "y":
    dvc = "gpu"
else:
    dvc ="cpu"

if an3 == "y":
    wns = "True"
else:
    wns ="False"



print("please wait for camera to start ")
cam = cv2.VideoCapture(0)

cv2.namedWindow("test")
```

```
img_counter = 0

while True:
    ret, frame = cam.read()
    if not ret:
        print("failed to grab frame")
        break
    cv2.imshow("test", frame)

    k = cv2.waitKey(1)
    if k%256 == 27:#esc
        print("Escape hit, closing...")
        break
    else:
        img_name = "opencv_frame_{}.png".format(img_counter)
        cv2.imwrite(img_name, frame)
        print("{} written!".format(img_name))
        img =
cv2.imread('C:/Users/masdaq/Desktop/ocv/ultralytics-main/opencv_
frame_'+str(img_counter)+'.png')
        #cv2.imshow('sample image', img)
        img_counter += 1
        #time.sleep(1)
    if img_counter == 5:
        img_counter = 0
        blur_finder()

cam.release()

cv2.destroyAllWindows()

#blur_finder()
```

References

https://worldpopulationreview.com/country-rankings/missing-persons-statistics-by-country

https://ieeexplore.ieee.org/abstract/document/7894491?casa_token=wv4ZaEN5_UQAAAAA:9Nx8Bx4Icpt3anl5XdnEuotUfGvpszvqmj7tRIPi016JtiBDJu0pJ4InUiimxC_KHLj_ls1gWfU

https://github.com/ultralytics/ultralytics

https://ieeexplore.ieee.org/abstract/document/9352144?casa_token=HTnwvzIco4cAAAAA:zYMCIcmKWX8j3KaMNP-6-wumMM1wBafPxBPG1b5xDhhuOsjcylrzG4fM2J_JWEZRDbJchYJWTPc

https://www.stereolabs.com/blog/performance-of-yolo-v5-v7-and-v8/

https://medium.com/mlearning-ai/yolo-v8-the-real-state-of-the-art-eda6c86a1b90 (Bhalerao)