Codes

- **200001006**
- **200005040**

Detect_mask_video.py Code-

detect_mask_video.py 6 × social_distance_detection.py 8 train_mask_detector.py 9+

C: > Users > DELL > Documents > face_mask > one > detect_mask_video.py > ...

```python
# import the necessary packages
from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
from tensorflow.keras.preprocessing.image import img_to_array
from tensorflow.keras.models import load_model
from imutils.video import VideoStream
import numpy as np
import imutils
import time
import cv2
import os

def detect_and_predict_mask(frame, faceNet, maskNet):
    # grab the dimensions of the frame and then construct a blob from it
    (h, w) = frame.shape[:2]
    blob = cv2.dnn.blobFromImage(frame, 1.0, (224, 224), (104.0, 177.0, 123.0))

    # pass the blob through the network and obtain the face detections
    faceNet.setInput(blob)    # setting input for facenet network
    detections = faceNet.forward()   # The detections list contains the predicted bounding boxes and confidence scores for each detected face
    print(detections.shape)

    # initialize our list of faces, their corresponding locations, and the list of predictions from our face mask network
    faces = []
    locs = []
    preds = []

    # loop over the detections
    for i in range(0, detections.shape[2]):
        # extract the confidence (i.e., probability) associated with the detection
        confidence = detections[0, 0, i, 2]

        # filter out weak detections by ensuring the confidence is
        # greater than the minimum confidence
        if confidence > 0.5:
            # compute the (x, y)-coordinates of the bounding box for
            # the object
            box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
            (startX, startY, endX, endY) = box.astype("int")

            # ensure the bounding boxes fall within the dimensions of
```

n Testcases ⊗ 0 ⚠ 28 Ln 6, Col 19 Tab Size: 4 UTF-8 LF {} Python

File Edit Selection View Go Run Terminal Help detect_ma

detect_mask_video.py 6 | social_distance_detection.py 8 | train_mask_detector.py 9+

C: > Users > DELL > Documents > face_mask > one > detect_mask_video.py > detect_and_predict_mask

```python
            # ensure the bounding boxes fall within the dimensions of
            # the frame
            (startX, startY) = (max(0, startX), max(0, startY))
            (endX, endY) = (min(w - 1, endX), min(h - 1, endY))

            # extract the face ROI, convert it from BGR to RGB channel
            # ordering, resize it to 224x224, and preprocess it
            face = frame[startY:endY, startX:endX]
            face = cv2.cvtColor(face, cv2.COLOR_BGR2RGB)
            face = cv2.resize(face, (224, 224))
            face = img_to_array(face)
            face = preprocess_input(face)

            # add the face and bounding boxes to their respective
            # lists
            faces.append(face)
            locs.append((startX, startY, endX, endY))

    # only make a predictions if at least one face was detected
    if len(faces) > 0:
        # for faster inference we'll make batch predictions on *all*
        # faces at the same time rather than one-by-one predictions
        # in the above `for` loop
        faces = np.array(faces, dtype="float32")
        preds = maskNet.predict(faces, batch_size=32)

    # return a 2-tuple of the face locations and their corresponding
    # locations
    return (locs, preds)

# load our serialized face detector model from disk
prototxtPath = r"face_detector\deploy.prototxt"
weightsPath = r"face_detector\res10_300x300_ssd_iter_140000.caffemodel"
faceNet = cv2.dnn.readNet(prototxtPath, weightsPath)

# load the face mask detector model from disk
maskNet = load_model("mask_detector.model")

# initialize the video stream
```

Run Testcases ⊗ 0 ⚠ 28

File Edit Selection View Go Run Terminal Help detect_masl

detect_mask_video.py 6 × social_distance_detection.py 8 train_mask_detector.py 9+

C: > Users > DELL > Documents > face_mask > one > detect_mask_video.py > detect_and_predict_mask

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

# loop over the frames from the video stream
while True:
    # grab the frame from the threaded video stream and resize it
    # to have a maximum width of 400 pixels
    frame = vs.read()
    frame = imutils.resize(frame, width=400)

    # detect faces in the frame and determine if they are wearing a
    # face mask or not
    (locs, preds) = detect_and_predict_mask(frame, faceNet, maskNet)

    # loop over the detected face locations and their corresponding
    # locations
    for (box, pred) in zip(locs, preds):
        # unpack the bounding box and predictions
        (startX, startY, endX, endY) = box
        (mask, withoutMask) = pred

        # determine the class label and color we'll use to draw
        # the bounding box and text
        label = "Mask" if mask > withoutMask else "No Mask"
        color = (0, 255, 0) if label == "Mask" else (0, 0, 255)

        # include the probability in the label
        label = "{}: {:.2f}%".format(label, max(mask, withoutMask) * 100)

        # display the label and bounding box rectangle on the output frame
        cv2.putText(frame, label, (startX, startY - 10),
            cv2.FONT_HERSHEY_SIMPLEX, 0.45, color, 2)
        cv2.rectangle(frame, (startX, startY), (endX, endY), color, 2)

    # show the output frame
    cv2.imshow("Frame", frame)
    key = cv2.waitKey(1) & 0xFF

    if key == ord("q"):
        break
```

Run Testcases ⊗ 0 ⚠ 28

File Edit Selection View Go Run Terminal Help detect_ma

detect_mask_video.py 6 × social_distance_detection.py 8 train_mask_detector.py 9+

C: > Users > DELL > Documents > face_mask > one > detect_mask_video.py > detect_and_predict_mask

```python
# loop over the frames from the video stream
while True:
    # grab the frame from the threaded video stream and resize it
    # to have a maximum width of 400 pixels
    frame = vs.read()
    frame = imutils.resize(frame, width=400)

    # detect faces in the frame and determine if they are wearing a
    # face mask or not
    (locs, preds) = detect_and_predict_mask(frame, faceNet, maskNet)

    # loop over the detected face locations and their corresponding
    # locations
    for (box, pred) in zip(locs, preds):
        # unpack the bounding box and predictions
        (startX, startY, endX, endY) = box
        (mask, withoutMask) = pred

        # determine the class label and color we'll use to draw
        # the bounding box and text
        label = "Mask" if mask > withoutMask else "No Mask"
        color = (0, 255, 0) if label == "Mask" else (0, 0, 255)

        # include the probability in the label
        label = "{}: {:.2f}%".format(label, max(mask, withoutMask) * 100)

        # display the label and bounding box rectangle on the output frame
        cv2.putText(frame, label, (startX, startY - 10),
            cv2.FONT_HERSHEY_SIMPLEX, 0.45, color, 2)
        cv2.rectangle(frame, (startX, startY), (endX, endY), color, 2)

    # show the output frame
    cv2.imshow("Frame", frame)
    key = cv2.waitKey(1) & 0xFF

    if key == ord("q"):
        break

cv2.destroyAllWindows()
vs.stop()
```

Run Testcases ⊗ 0 ⚠ 28

Train_mask_detector.py Code-

File Edit Selection View Go Run Terminal Help train_mask_detector.py - Visual Studio Code

detect_mask_video.py 6 social_distance_detection.py 8 train_mask_detector.py 9+

C: > Users > DELL > Documents > face_mask > one > train_mask_detector.py > ...

```python
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator  # Data preprocesing, Augmentation
from tensorflow.keras.applications import MobileNetV2                # Base Network for our model
#Importing different layers for our model
from tensorflow.keras.layers import AveragePooling2D
from tensorflow.keras.layers import Dropout
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Input
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.applications.mobilenet_v2 import preprocess_input    # Scaling pixel intensities
from tensorflow.keras.preprocessing.image import img_to_array              # Conversion of image to array format
from tensorflow.keras.preprocessing.image import load_img                  # Resizing
# One Hot Encoding
from tensorflow.keras.utils import to_categorical
from sklearn.preprocessing import LabelBinarizer
# Spliting the testing and trainging data
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from imutils import paths
import matplotlib.pyplot as plt
import numpy as np
import os

# initializing the initial learning rate, number of epochs to train for and Batch size
INIT_LR = 1e-4
EPOCHS = 20
BS = 32

DIRECTORY = r"C:\one\dataset"
CATEGORIES = ["with_mask", "without_mask"]

print(" Please Wait till the Images are being loaded ... ")

# Data list to store the images and the Labels list to store the corresponding labels (mask ,unmasked)
data = []
labels = []

#Image Preprocessing before feeding the images to the model
```

Run Testcases ⊗ 0 ⚠ 28 Ln 12, Col

File Edit Selection View Go Run Terminal Help train_mask_detector.py - Visual Studio Code

detect_mask_video.py 6 | social_distance_detection.py 8 | train_mask_detector.py 9+

C: > Users > DELL > Documents > face_mask > one > train_mask_detector.py > ...

```python
#Image Preprocessing before feeding the images to the model
for category in CATEGORIES:
    path = os.path.join(DIRECTORY, category)                # Creating full pathe to the directory of the images
    for img in os.listdir(path):
        img_path = os.path.join(path, img)                  # Resizing the Images to 224x224 pixels ensuring all images
        image = load_img(img_path, target_size=(224, 224))  # Conversion to numpy array format
        image = img_to_array(image)                         # Scaling the pixel intensities in the input image to range
        image = preprocess_input(image)

data.append(image)          # Appending the images to the Data List
labels.append(category)     # Appending the corresponding label to the Labels List

# perform one-hot encoding on the labels
lb = LabelBinarizer()                # converting categorical data to binar vectors
labels = lb.fit_transform(labels)    # fits the label binarizar to labels and transforms the labels
labels = to_categorical(labels)      # convert vinary vectors to one hot encoded labels

# Numpy Array
data = np.array(data, dtype="float32")
labels = np.array(labels)

# Spliting the dataset into training and testing dataset
# Stratify ensures that each class in the target variable is proportionally represented in the training and testing sets.
(trainX, testX, trainY, testY) = train_test_split(data, labels, test_size=0.20, stratify=labels, random_state=42)

# construct the training image generator for data augmentation
aug = ImageDataGenerator(
    rotation_range=20,      # Rotation
    zoom_range=0.15,        # Magnification
    width_shift_range=0.2,  # Horizontal Shift
    height_shift_range=0.2, # Vertical Shift
    shear_range=0.15,       # Shear
    horizontal_flip=True,   # Flip
    fill_mode="nearest")    # Fill empty pixels with the nearest pixels on shifting

# load the MobileNetV2 network, Excluding the fully connected layer to nuild our own binary classification

baseModel = MobileNetV2(weights="imagenet", include_top=False, input_tensor=Input(shape=(224, 224, 3)))

# construct the head of the model  to be placed on the head of the base model
```

Run Testcases ⊗ 0 ⚠ 28 Ln 12, Col 103 Tab Si

File Edit Selection View Go Run Terminal Help train_mask_detector.py - Visual

detect_mask_video.py 6 | social_distance_detection.py 8 | train_mask_detector.py 9+

C: > Users > DELL > Documents > face_mask > one > train_mask_detector.py > ...

```python
# construct the head of the model , to be placed on the head of the base model
headModel = baseModel.output
headModel = AveragePooling2D(pool_size=(7, 7))(headModel)
headModel = Flatten(name="flatten")(headModel)
headModel = Dense(128, activation="relu")(headModel)
headModel = Dropout(0.5)(headModel)
headModel = Dense(2, activation="softmax")(headModel)

# The fully Connected model is placed on the top of the base model to get our actual model
model = Model(inputs=baseModel.input, outputs=headModel)

# loop over all layers in the base model and freeze them so they wont be updated during the
# because the pre-trained MobileNetV2 model has already learned important features from the
for layer in baseModel.layers:
    layer.trainable = False

# compile our model with Accuracy metrics for evaluation of model and binary crossentropy
print("[INFO] compiling model...")
opt = tf.keras.optimizers.legacy.Adam(lr=INIT_LR, decay=INIT_LR / EPOCHS)     #Adam optimiz
model.compile(loss="binary_crossentropy", optimizer=opt, metrics=["accuracy"])

# train the head of the network using the augmented data , providing steps per epoch and ba
print("[INFO] training head...")
H = model.fit(
    aug.flow(trainX, trainY, batch_size=BS),
    steps_per_epoch=len(trainX) // BS,
    validation_data=(testX, testY),
    validation_steps=len(testX) // BS,
    epochs=EPOCHS)

# make predictions on the testing set
print("[INFO] evaluating network...")
predIdxs = model.predict(testX, batch_size=BS)

# for each image in the testing set we need to find the index of the
# label with corresponding largest predicted probability
predIdxs = np.argmax(predIdxs, axis=1)

# show a nicely formatted classification report
```

Run Testcases ⊗ 0 ⚠ 28

File Edit Selection View Go Run Terminal Help train_mask_detector.py - V

detect_mask_video.py 6 | social_distance_detection.py 8 | train_mask_detector.py 9+

C: > Users > DELL > Documents > face_mask > one > train_mask_detector.py > ...

```python
print("[INFO] training head...")
H = model.fit(
    aug.flow(trainX, trainY, batch_size=BS),
    steps_per_epoch=len(trainX) // BS,
    validation_data=(testX, testY),
    validation_steps=len(testX) // BS,
    epochs=EPOCHS)

# make predictions on the testing set
print("[INFO] evaluating network...")
predIdxs = model.predict(testX, batch_size=BS)

# for each image in the testing set we need to find the index of the
# label with corresponding largest predicted probability
predIdxs = np.argmax(predIdxs, axis=1)

# show a nicely formatted classification report
print(classification_report(testY.argmax(axis=1), predIdxs, target_names=lb.classes_))

# serialize the model to disk
print("[INFO] saving mask detector model...")
model.save("mask_detector.model", save_format="h5")

# plot the training loss and accuracy
N = EPOCHS
plt.style.use("ggplot")
plt.figure()
plt.plot(np.arange(0, N), H.history["loss"], label="train_loss")
plt.plot(np.arange(0, N), H.history["val_loss"], label="val_loss")
plt.plot(np.arange(0, N), H.history["accuracy"], label="train_acc")
plt.plot(np.arange(0, N), H.history["val_accuracy"], label="val_acc")
plt.title("Training Loss and Accuracy")
plt.xlabel("Epoch #")
plt.ylabel("Loss/Accuracy")
plt.legend(loc="lower left")
plt.savefig("plot.png")
```

Run Testcases ⊗ 0 ⚠ 28

Social_mask_distancing.py Code-

File Edit Selection View Go Run Terminal Help social_di

detect_mask_video.py 6 | social_distance_detection.py 8 | train_mask_detector.py 9+

C: > Users > DELL > Documents > face_mask > one > social_distance_detection.py > ...

```python
from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
from tensorflow.keras.preprocessing.image import img_to_array
from tensorflow.keras.models import load_model
from imutils.video import VideoStream
import numpy as np
import argparse
import imutils
import time
import cv2
import os
import sys
import cv2
from math import pow, sqrt
import playsound
from threading import Thread

def detect_and_predict_mask(frame, faceNet, maskNet):
    # grab the dimensions of the frame and then construct a blob
    # from it
    (h, w) = frame.shape[:2]
    blob = cv2.dnn.blobFromImage(frame, 1.0, (300, 300),
        (104.0, 177.0, 123.0))

    # pass the blob through the network and obtain the face detections
    faceNet.setInput(blob)
    detections = faceNet.forward()

    # initialize our list of faces, their corresponding locations,
    # and the list of predictions from our face mask network
    faces = []
    locs = []
    preds = []

    # loop over the detections
    for i in range(0, detections.shape[2]):
        # extract the confidence (i.e., probability) associated with
        # the detection
        confidence = detections[0, 0, i, 2]

        # filter out weak detections by ensuring the confidence is
```

Run Testcases ⊗ 0 ⚠ 28

File Edit Selection View Go Run Terminal Help social_dis

detect_mask_video.py 6 | social_distance_detection.py 8 | train_mask_detector.py 9+

C: > Users > DELL > Documents > face_mask > one > social_distance_detection.py > ...

```python
        # filter out weak detections by ensuring the confidence is
        # greater than the minimum confidence
        if confidence > args["confidence"]:
            # compute the (x, y)-coordinates of the bounding box for
            # the object
            box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
            (startX, startY, endX, endY) = box.astype("int")

            # ensure the bounding boxes fall within the dimensions of
            # the frame
            (startX, startY) = (max(0, startX), max(0, startY))
            (endX, endY) = (min(w - 1, endX), min(h - 1, endY))

            # extract the face ROI, convert it from BGR to RGB channel
            # ordering, resize it to 224x224, and preprocess it
            face = frame[startY:endY, startX:endX]
            face = cv2.cvtColor(face, cv2.COLOR_BGR2RGB)
            face = cv2.resize(face, (224, 224))
            face = img_to_array(face)
            face = preprocess_input(face)
            face = np.expand_dims(face, axis=0)

            # add the face and bounding boxes to their respective
            # lists
            faces.append(face)
            locs.append((startX, startY, endX, endY))

    # only make a predictions if at least one face was detected
    if len(faces) > 0:
        # for faster inference we'll make batch predictions on *all*
        # faces at the same time rather than one-by-one predictions
        # in the above `for` loop
        preds = maskNet.predict(faces)

    # return a 2-tuple of the face locations and their corresponding
    # locations
    return (locs, preds)


# Parse the arguments from command line
```

Run Testcases ⊗ 0 ⚠ 28

it Selection View Go Run Terminal Help social_distance_detection.py - Visual Studio Code

tect_mask_video.py 6 | social_distance_detection.py 8 × | train_mask_detector.py 9+

Users > DELL > Documents > face_mask > one > social_distance_detection.py > ...

```python
# Parse the arguments from command line
arg = argparse.ArgumentParser(description='Social distance detection')

arg.add_argument("-a", "--alarm", type=str, default="", help="path alarm .WAV file")

arg.add_argument('-v', '--video', type = str, default = '', help = 'Video file path. If no path is given, video is captured using device.')

arg.add_argument('-m', '--model', required = True, help = "Path to the pretrained model.")

arg.add_argument('-p', '--prototxt', required = True, help = 'Prototxt of the model.')

arg.add_argument('-l', '--labels', required = True, help = 'Labels of the dataset.')

arg.add_argument('-c', '--confidence', type = float, default = 0.2, help='Set confidence for detecting objects')

arg.add_argument("-f", "--face", type=str, default="face_detector", help="path to face detector model directory")

arg.add_argument("-m1", "--model1", type=str, default="model/mask_detector.model", help="path to trained face mask detector model")

#arg.add_argument("-c", "--confidence", type=float, default=0.5,    help="minimum probability to filter weak detections")

args = vars(arg.parse_args())

ALARM_ON = False

def sound_alarm(path):
    # play an alarm sound
    playsound.playsound(path)

labels = [line.strip() for line in open(args['labels'])]

# Generate random bounding box bounding_box_color for each label
bounding_box_color = np.random.uniform(0, 255, size=(len(labels), 3))

print("[INFO] loading face detector model...")

prototxtPath = os.path.sep.join([args["face"], "deploy.prototxt"])

weightsPath = os.path.sep.join([args["face"], "res10_300x300_ssd_iter_140000.caffemodel"])
```

ses ⊗ 0 ⚠ 28 Ln 12, Col 11 Spaces: 4 UTF-8 CRLF {} Py

File Edit Selection View Go Run Terminal Help

detect_mask_video.py 6 | social_distance_detection.py 8 | train_mask_detector.py 9

C: > Users > DELL > Documents > face_mask > one > social_distance_detection.py > ...

```python
faceNet = cv2.dnn.readNet(prototxtPath, weightsPath)

# load the face mask detector model from disk
print("[INFO] loading face mask detector model...")
maskNet = load_model(args["model1"])

# Load model
print("\nLoading model...\n")
network = cv2.dnn.readNetFromCaffe(args["prototxt"], args["model"])

print("\nStreaming video using device...\n")


# Capture video from file or through device
if args['video']:
    cap = cv2.VideoCapture(args['video'])
else:
    cap = cv2.VideoCapture(0)


frame_no = 0

while True:

    frame_no = frame_no+1

    # Capture one frame after another
    ret, frame = cap.read()

    #frame = imutils.resize(frame, width=1000)

    # detect faces in the frame and determine if they are wearing a

    if not ret:
        break

    (h, w) = frame.shape[:2]

    # Resize the frame to suite the model requirements. Resize the fra
```

Run Testcases ⊗ 0 ⚠ 28

File Edit Selection View Go Run Terminal Help social_distance_detection.py - Visual

detect_mask_video.py 6 | social_distance_detection.py 8 | train_mask_detector.py 9+

C: > Users > DELL > Documents > face_mask > one > social_distance_detection.py > ...

```python

    # Resize the frame to suite the model requirements. Resize the frame to 300X300 pixels
    blob = cv2.dnn.blobFromImage(cv2.resize(frame, (300, 300)), 0.007843, (300, 300), 127.5)

    network.setInput(blob)
    detections = network.forward()

    pos_dict = dict()
    coordinates = dict()

    # Focal length
    F = 615

    for i in range(detections.shape[2]):

        confidence = detections[0, 0, i, 2]

        if confidence > args["confidence"]:

            class_id = int(detections[0, 0, i, 1])

            box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
            (startX, startY, endX, endY) = box.astype('int')

            # Filtering only persons detected in the frame. Class Id of 'person' is 15
            if class_id == 15.00:

                (locs, preds) = detect_and_predict_mask(frame, faceNet, maskNet)

                # loop over the detected face locations and their corresponding
                # locations
                for (box, pred) in zip(locs, preds):
                    # unpack the bounding box and predictions
                    (startX_mask, startY_mask, endX_mask, endY_mask) = box
                    (mask, withoutMask) = pred

                    # determine the class label and color we'll use to draw
                    # the bounding box and text
                    if mask > withoutMask:
                        label_mask = "Mask"
```

Run Testcases ⊗ 0 ⚠ 28

File Edit Selection View Go Run Terminal Help social_distance_detection.py - Visual Studio Code

detect_mask_video.py 6 | social_distance_detection.py 8 | train_mask_detector.py 9+

C: > Users > DELL > Documents > face_mask > one > social_distance_detection.py > ...

```python
                label_mask = "Mask"
                color = (0, 255, 0)
                ALARM_ON = False
            else:
                label_mask="No Mask"
                color = (0, 0, 255)
                if not ALARM_ON:
                    ALARM_ON = True
                    if args["alarm"] != "":
                        t = Thread(target=sound_alarm, args=(args["alarm"],))
                        t.deamon = True
                        t.start()

        # Draw bounding box for the object
        cv2.rectangle(frame, (startX, startY), (endX, endY), bounding_box_color[class_id], 2)

        label = "{}: {:.2f}%".format(labels[class_id], confidence * 100)
        label_mask = "{}: {:.2f}%".format(label_mask, max(mask, withoutMask) * 100)
        print("{}".format(label))
        cv2.putText(frame, label, (startX_mask, startY_mask - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.45, color, 2)
        cv2.rectangle(frame, (startX_mask, startY_mask), (endX_mask, endY_mask), color, 2)


        coordinates[i] = (startX, startY, endX, endY)

        # Mid point of bounding box
        x_mid = round((startX+endX)/2,4)
        y_mid = round((startY+endY)/2,4)

        height = round(endY-startY,4)

        # Distance from camera based on triangle similarity
        distance = (165 * F)/height
        print("Distance(cm):{dist}\n".format(dist=distance))

        # Mid-point of bounding boxes (in cm) based on triangle similarity technique
        x_mid_cm = (x_mid * distance) / F
        y_mid_cm = (y_mid * distance) / F
        pos_dict[i] = (x_mid_cm,y_mid_cm,distance)
```

Run Testcases ⊗ 0 ⚠ 28 Ln 12, Col 11

File Edit Selection View Go Run Terminal Help social_distance_detection.py - Visual Studio Code

detect_mask_video.py 6 | social_distance_detection.py 8 | train_mask_detector.py 9+

C: > Users > DELL > Documents > face_mask > one > social_distance_detection.py > ...

```python
    # Distance between every object detected in a frame
    close_objects = set()
    for i in pos_dict.keys():
        for j in pos_dict.keys():
            if i < j:
                dist = sqrt(pow(pos_dict[i][0]-pos_dict[j][0],2) + pow(pos_dict[i][1]-pos_dict[j][1],2) + pow(pos_dict[i][2]-pos_dict[j][2],2))

                # Check if distance less than 2 metres or 200 centimetres
                if dist < 200:
                    close_objects.add(i)
                    close_objects.add(j)

    for i in pos_dict.keys():
        if i in close_objects:
            COLOR = (0,0,255)
            if not ALARM_ON:
                ALARM_ON = True
                if args["alarm"] != "":
                    t = Thread(target=sound_alarm, args=(args["alarm"],))
                    t.deamon = True
                    t.start()

        else:
            COLOR = (0,255,0)
            ALARM_ON = False
        (startX, startY, endX, endY) = coordinates[i]

        cv2.rectangle(frame, (startX, startY), (endX, endY), COLOR, 2)
        y = startY - 15 if startY - 15 > 15 else startY + 15
        # Convert cms to feet
        cv2.putText(frame, 'Depth: {i} ft'.format(i=round(pos_dict[i][2]/30.48,4)), (startX, y),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.5, COLOR, 2)

    cv2.namedWindow('Frame',cv2.WINDOW_NORMAL)

    # Show frame
    cv2.imshow('Frame', frame)
    cv2.resizeWindow('Frame',800,600)
```

Run Testcases ⊗ 0 ⚠ 28 Ln 12, Col 11 Spaces: 4 UTF-8 CRLF {} Python