Deploy_UI

August 19, 2020

[ ]:
```
### This program is to create User interface to upload videos/images from folder
 ↪or camera and run C2TSR model on it.

import numpy
from pygame import mixer
import time
import cv2
import filetype
import numpy as np
from tkinter import *
from tkinter import filedialog
import tkinter.messagebox
from datetime import datetime
import mimetypes

root = Tk()
root.geometry('500x570')
frame = Frame(root, relief=RIDGE, borderwidth=2)
frame.pack(fill=BOTH, expand=1)
root.title('C2TSR')
frame.config(background='light blue')
label = Label(frame, text="C2TSR: Concurrent Canada-based Traffic Signposts
 ↪Recognition System", bg='light blue', font=('Times 12 bold'))
label.pack(side=TOP)
label1 = Label(frame, text=datetime.now(), bg='light blue', font=('Times 12
 ↪bold'))
label1.pack(side=BOTTOM)

label2 = Label(frame, text="Select a video or image file", bg='light blue',
 ↪font=('Times 10'))
label2.place(x=10, y=100)

label3 = Label(frame, text="", bg='light blue', font=('Times 10'))
label3.place(x=10, y=120)
#label3.pack()

def hel():
```

```
    help(cv2)


def Contri():
    tkinter.messagebox.showinfo("Contributors", "\n1.Suby Singh \n")


def anotherWin():
    tkinter.messagebox.showinfo("About",
                                'C2TSR version v1.0\n Model trained␣
 ↪using\n-Darknet Deep learning framework\n-YOLO algorithm\nMade␣
 ↪Using\n-OpenCV\n-Numpy\n-Tkinter\n In Python 3')


menu = Menu(root)
root.config(menu=menu)

subm1 = Menu(menu)
menu.add_cascade(label="Tools", menu=subm1)
subm1.add_command(label="Open CV Docs", command=hel)

subm2 = Menu(menu)
menu.add_cascade(label="About", menu=subm2)
subm2.add_command(label="C2TSR", command=anotherWin)
subm2.add_command(label="Contributors", command=Contri)
file_path=""
def exitt():
    exit()

def run():
    global file_path
    # Dialog box to input image
    file_path = filedialog.askopenfilename()
    print(file_path)
    label3.configure(text=file_path)
    kind = filetype.guess(file_path)
    if kind is None:
        print('Cannot guess file type!')
        tkinter.messagebox.showerror(title=None, message="Invalid file type.\n␣
 ↪Please select a video or image.")
        return
    mimetypes.init()
    mimestart = mimetypes.guess_type(file_path)[0]
    if mimestart != None:
        mimestart = mimestart.split('/')[0]
        if mimestart == 'image':
            run_image()
```

```
        if mimestart == 'video':
            run_video()

def run_video():
    #Load the classes
    classes = []
    path = r'..\miscFiles\FinalC2TSR.names'
    with open(path, "r") as f:
        classes = [line.strip() for line in f.readlines()]
    # loading model and configuration files
    net = cv2.dnn.readNet("../Model/yolov3_custom_last.weights", "../miscFiles/
 ↪yolov3_custom.cfg")
    # get layers
    layer_names = net.getLayerNames()
    # get output layer
    output_layers = [layer_names[i[0] - 1] for i in net.
 ↪getUnconnectedOutLayers()]
    colors = numpy.random.uniform(0, 255, size=(len(classes), 3))
    #Dialog box to input video
    file_name = label3["text"]
    #print(file_path)
    # input video
    cap = cv2.VideoCapture(file_name)
    # set font of the text
    font = cv2.FONT_HERSHEY_COMPLEX_SMALL
    starting_time = time.time()
    frame_id = 0
    windowname="C2TSR-video"
    while True:
        # read frames
        _, frame = cap.read()
        frame = cv2.resize(frame, None, fx=0.6, fy=0.6)
        frame_id += 1

        height, width, channels = frame.shape

        # Detecting objects
        blob = cv2.dnn.blobFromImage(frame, 0.00392, (416, 416), (0, 0, 0),␣
 ↪True, crop=False)

        net.setInput(blob)
        outs = net.forward(output_layers)

        # Showing informations on the screen
        class_ids = []
        confidences = []
        boxes = []
```

```
        for out in outs:
            for detection in out:
                scores = detection[5:]
                class_id = np.argmax(scores)
                confidence = scores[class_id]
                if confidence > 0.3:
                    # Object detected
                    center_x = int(detection[0] * width)
                    center_y = int(detection[1] * height)
                    w = int(detection[2] * width)
                    h = int(detection[3] * height)

                    # Rectangle coordinates
                    x = int(center_x - w / 2)
                    y = int(center_y - h / 2)

                    boxes.append([x, y, w, h])
                    confidences.append(float(confidence))
                    class_ids.append(class_id)
        #Decide the boundary boxes for multiple detected signs
        indexes = cv2.dnn.NMSBoxes(boxes, confidences, 0.8, 0.3)

        for i in range(len(boxes)):
            if i in indexes:
                x, y, w, h = boxes[i]
                label = str(classes[class_ids[i]])
                confidence = confidences[i]
                color = colors[class_ids[i]]
                #Display the detected signs
                cv2.rectangle(frame, (x, y), (x + w, y + h), color, 2)
                cv2.putText(frame, label + " " + str(round(confidence, 2)), (x,␣
↪y), font, 0.5, color, 1)

        elapsed_time = time.time() - starting_time
        #display the FPS value
        fps = frame_id / elapsed_time
        cv2.putText(frame, "FPS: " + str(round(fps, 2)), (10, 50), font, 1, (0,␣
↪0, 0), 3)

        cv2.imshow(windowname, frame)
        #key = cv2.waitKey(1)
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
    #Free up memory
    cap.release()
    cv2.destroyAllWindows()
```

```
def run_image():
    global file_path
    #Load classes
    classes = []
    path = r'..\miscFiles\FinalC2TSR.names'
    with open(path, "r") as f:
        classes = [line.strip() for line in f.readlines()]

    # loading model and configuration files
    net = cv2.dnn.readNet("../Model/yolov3_custom_last.weights", "../miscFiles/
↪yolov3_custom.cfg")
    # get layers
    layer_names = net.getLayerNames()
    # get output layer
    output_layers = [layer_names[i[0] - 1] for i in net.
↪getUnconnectedOutLayers()]
    colors = numpy.random.uniform(0, 255, size=(len(classes), 3))

    file_name = label3["text"]
    #print(file_path)
    # input image
    frame = cv2.imread(file_name)
    #set the text font
    font = cv2.FONT_HERSHEY_COMPLEX_SMALL
    starting_time = time.time()
    frame = cv2.resize(frame, None, fx=0.5, fy=0.5)
    height, width, channels = frame.shape

    # Detecting objects
    blob = cv2.dnn.blobFromImage(frame, 0.00392, (416, 416), (0, 0, 0), True,␣
↪crop=False)

    net.setInput(blob)
    outs = net.forward(output_layers)

    # Showing informations on the screen
    class_ids = []
    confidences = []
    boxes = []
    for out in outs:
        for detection in out:
            scores = detection[5:]
            class_id = np.argmax(scores)
            confidence = scores[class_id]
            if confidence > 0.3:
                # Object detected
                center_x = int(detection[0] * width)
```

```
                center_y = int(detection[1] * height)
                w = int(detection[2] * width)
                h = int(detection[3] * height)

                # Rectangle coordinates
                x = int(center_x - w / 2)
                y = int(center_y - h / 2)

                boxes.append([x, y, w, h])
                confidences.append(float(confidence))
                class_ids.append(class_id)
    # Decide the boundary boxes for multiple detected signs
    indexes = cv2.dnn.NMSBoxes(boxes, confidences, 0.8, 0.3)

    for i in range(len(boxes)):
        if i in indexes:
            x, y, w, h = boxes[i]
            label = str(classes[class_ids[i]])
            confidence = confidences[i]
            color = colors[class_ids[i]]
            #Display the detected traffic signs
            cv2.rectangle(frame, (x, y), (x + w, y + h), color, 2)
            cv2.putText(frame, label + " " + str(round(confidence, 2)), (x, y),␣
↪font, 0.5, color, 1)

    elapsed_time = time.time() - starting_time
    cv2.imshow("C2TSR-image", frame)
    key = cv2.waitKey(0)
    cv2.destroyAllWindows()

def run_camera():
    #Load the classes
    classes = []
    path = r'..\miscFiles\FinalC2TSR.names'
    with open(path, "r") as f:
        classes = [line.strip() for line in f.readlines()]
    # loading model and configuration files
    net = cv2.dnn.readNet("../Model/yolov3_custom_last.weights", "../miscFiles/
↪yolov3_custom.cfg")
    # get layers
    layer_names = net.getLayerNames()
    # get output layer
    output_layers = [layer_names[i[0] - 1] for i in net.
↪getUnconnectedOutLayers()]
    colors = numpy.random.uniform(0, 255, size=(len(classes), 3))

    # input video
```

```
    cap = cv2.VideoCapture(0)
    # set font of the text
    font = cv2.FONT_HERSHEY_COMPLEX_SMALL
    starting_time = time.time()
    frame_id = 0
    windowname="C2TSR-video"
    while True:
        # read frames
        _, frame = cap.read()
        frame = cv2.resize(frame, None, fx=0.8, fy=0.8)
        frame_id += 1

        height, width, channels = frame.shape

        # Detecting objects
        blob = cv2.dnn.blobFromImage(frame, 0.00392, (416, 416), (0, 0, 0),␣
↪True, crop=False)

        net.setInput(blob)
        outs = net.forward(output_layers)

        # Showing informations on the screen
        class_ids = []
        confidences = []
        boxes = []
        for out in outs:
            for detection in out:
                scores = detection[5:]
                class_id = np.argmax(scores)
                confidence = scores[class_id]
                if confidence > 0.3:
                    # Object detected
                    center_x = int(detection[0] * width)
                    center_y = int(detection[1] * height)
                    w = int(detection[2] * width)
                    h = int(detection[3] * height)

                    # Rectangle coordinates
                    x = int(center_x - w / 2)
                    y = int(center_y - h / 2)

                    boxes.append([x, y, w, h])
                    confidences.append(float(confidence))
                    class_ids.append(class_id)
        #Decide the boundary boxes for multiple detected signs
        indexes = cv2.dnn.NMSBoxes(boxes, confidences, 0.8, 0.3)
```

```
        for i in range(len(boxes)):
            if i in indexes:
                x, y, w, h = boxes[i]
                label = str(classes[class_ids[i]])
                confidence = confidences[i]
                color = colors[class_ids[i]]
                #Display the detected signs
                cv2.rectangle(frame, (x, y), (x + w, y + h), color, 2)
                cv2.putText(frame, label + " " + str(round(confidence, 2)), (x,␣
 ↪y), font, 0.5, color, 1)

        elapsed_time = time.time() - starting_time
        #display the FPS value
        fps = frame_id / elapsed_time
        cv2.putText(frame, "FPS: " + str(round(fps, 2)), (10, 50), font, 1, (0,␣
 ↪0, 0), 3)

        cv2.imshow(windowname, frame)
        #key = cv2.waitKey(1)
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
    #Free up memory
    cap.release()
    cv2.destroyAllWindows()

but1 = Button(frame, padx=5, pady=5, width=10, bg='white', fg='black',␣
 ↪relief=GROOVE, command=run, text='Browse',
              font=('helvetica 10 bold'))
but1.place(x=10, y=150)

but2 = Button(frame, padx=5, pady=5, width=10, bg='white', fg='black',␣
 ↪relief=GROOVE, command=run_camera, text='Open Camera',
              font=('helvetica 10 bold'))
but2.place(x=10,y=250)

but3 = Button(frame, padx=5, pady=5, width=5, bg='white', fg='black',␣
 ↪relief=GROOVE, text='EXIT', command=exitt,
              font=('helvetica 12 bold'))
but3.place(x=210, y=478)

root.mainloop()
```