## Appendix 1 – The full source code

An electronic version of the source code with additional configuration files is Available from https://github.com/pinojoke/ST-Open-Article-Detection-and-Classification-of-Cars-.

```
import os
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'
import time
import cv2
import numpy as np
from color_recognition_api import color_histogram_feature_extraction
from color_recognition_api import knn_classifier
import pytesseract
import tensorflow as tf
tf.get_logger().setLevel('ERROR')
import concurrent.futures
import glob
import PIL
import imutils
#detekcija automobila - yolo
net = cv2.dnn.readNetFromDarknet("yolo/yolov4-tiny_custom.cfg",
"yolo/yolov4-tiny_custom.weights")
layer_names = net.getLayerNames()
output_layers = [layer_names[i[0] - 1] for i in
net.getUnconnectedOutLayers()]
yolo_class = ['Automobil']

#funkcija klasifikacija boje
def color(image):
    prediction = ''
  color_histogram_feature_extraction.color_histogram_of_test_image(image)
    prediction = knn_classifier.main('training.data', 'test.data')
    #print(prediction)
    cv2.putText(frame, "Boja: " + str(prediction), (20,25),
cv2.FONT_HERSHEY_PLAIN, 2, (0,0,0), 2)
    return prediction
#detekcija i citanje tablice
custom_oem_psm_config = r'-l hrv --oem 1 --psm 3 -c
tessedit_char_whitelist=0123456789ABCČĆDEFGHIJKLMNOPRSŠTUVZŽ'
def licence_plate(image):
    image = imutils.resize(image, width=500)
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    gray = cv2.bilateralFilter(gray, 13, 15, 15)
    edged = cv2.Canny(gray, 30, 200)
    #trazenje kontura
    (cnts, _) = cv2.findContours(edged.copy(), cv2.RETR_LIST,
cv2.CHAIN_APPROX_SIMPLE)
    cnts = sorted(cnts, key = cv2.contourArea, reverse = True)[:10]
    for c in cnts:
```

```
            peri = cv2.arcLength(c, True)
            approx = cv2.approxPolyDP(c, 0.018 * peri, True)
            if len(approx) == 4:
                NumberPlateCnt = approx
                break
    mask = np.zeros(gray.shape,np.uint8)
    new_image = cv2.drawContours(mask,[NumberPlateCnt],0,255,-1,)
    new_image = cv2.bitwise_and(image,image,mask=mask)
    (x, y) = np.where(mask == 255)
    (topx, topy) = (np.min(x), np.min(y))
    (bottomx, bottomy) = (np.max(x), np.max(y))
    # izdvojena tablica
    Cropped = new_image[topx:bottomx+1, topy:bottomy+1]
    gray_crop = cv2.cvtColor(new_image, cv2.COLOR_BGR2GRAY)
    gray_crop = cv2.bilateralFilter(gray_crop, 13, 15, 15)
    cv2.imshow("tablica", gray_crop)
    #citanje znakova sa tablice
    text = pytesseract.image_to_string(gray_crop,
config=custom_oem_psm_config)
    cv2.putText(frame,"Tablica: " + str(text), (20,65),
cv2.FONT_HERSHEY_PLAIN, 2, (0,0,0), 2)
    print(text)
    return new_image
#klasifikacija modela automobila - ssd
interpreter_model = tf.lite.Interpreter(model_path='lite/model.tflite')
with open('lite/label_map.txt', 'r') as f:
    labels = [line.strip() for line in f.readlines()]
interpreter_model.allocate_tensors()
input_details = interpreter_model.get_input_details()
output_details = interpreter_model.get_output_details()

def model(image):
    height = input_details[0]['shape'][1]
    width = input_details[0]['shape'][2]
    object_name = ''
    frame_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    frame_resized = cv2.resize(frame_rgb, (width, height))
    input_data = np.expand_dims(frame_resized, axis=0)
    input_data = (np.float32(input_data) - 127.5) / 127.5
    interpreter_model.set_tensor(input_details[0]['index'],input_data)
    interpreter_model.invoke()
    classes = interpreter_model.get_tensor(output_details[1]['index'])[0]
    scores = interpreter_model.get_tensor(output_details[2]['index'])[0]
    for i in range(len(scores)):
    #samo ako je vjerojatnost veća od 80%
        if (scores[i] > 0.8):
            object_name = labels[int(classes[i])]
    cv2.putText(frame,"Model: " + str(object_name), (20,105),
cv2.FONT_HERSHEY_PLAIN, 2, (0,0,0), 2)
```

```
    return object_name

vid = cv2.VideoCapture('video/vitara.mp4')
if __name__ == '__main__':
    while True:
        height, width, channels = frame.shape
        #Ulaz u CNN
        blob = cv2.dnn.blobFromImage(frame, 0.00392, (96, 96), (0, 0, 0),
True, crop=False)
        net.setInput(blob)
        outs = net.forward(output_layers)
        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 > 0.8:
                    center_x = int(detection[0] * width)
                    center_y = int(detection[1] * height)
                    w = int(detection[2] * width)
                    h = int(detection[3] * height)
                    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)
        indexes = cv2.dnn.NMSBoxes(boxes, confidences, 0.5, 0.4)
        for i in range(len(boxes)):
            if i in indexes:
                x, y, w, h = boxes[i]
                label = str(yolo_class[class_ids[i]])
                #crtanje graničnog okvira
                cv2.rectangle(frame, (x,y), (x+w,y+h),(46,139,87), 2)
                car_crop=frame[int(y+10):int(y+h-10),int(x+10):int(x+w-
10)]
                height, width, channels = car.shape
                start_row, start_col = int(height * .5), int(0)
                end_row, end_col = int(height), int(width)
                color_crop=car_crop[start_row:end_row,start_col:end_col]
                #poziv funkcija
                color(color_crop)
                licence_plate(car_crop)
                model(car_crop)
        cv2.imshow("Klasifikacija_automobila", frame)
        if cv2.waitKey(1) & 0xFF == ord('q'): break
    cv2.destroyAllWindows()
```