Source Code Program ***Sistem Keamanan Gerbang Parkir Menggunakan Algoritma YOLO (You Only Look Once) dan Face Recognition***

Penulis: Kemal Taufik Fikri, Tata Supriyadi

Tanggal: 30 April 2022

1. Encoding wajah

Kode program fungsi untuk melakukan encoding wajah

```
import face_recognition as fr
import cv2
def findEncoding(image):
    img = cv2.cvtColor(image,cv2.COLOR_BGR2RGB)
    encode = fr.face_encodings(img)[0]
    return encode

def compareImages(encodeDisimpan,encodeBaru):
    results =
fr.compare_faces([encodeDisimpan],encodeBaru)
    faceDis =
fr.face_distance([encodeDisimpan],encodeBaru)

    return [results,faceDis]
```

2. Entrance gate

Kode utama pada gerbang masuk yang akan memanggil beberapa fungsi untuk menjalankan program seperti fungsi deteksi plat, pengenalan karakter, pengenalan wajah, dan query data

```
from yolobykemal import yolo
from imgpreprocesing import preprocessing
from encodingWajah import findEncoding
from gateRegister import simpanDataPintuMasuk
import cv2
```

```
from queryData import insertDatabase, insertPlat,
readData
import sys
import timeit

tic = timeit.default_timer()
#DETEKSI PLAT

## memasukkan parameter
weight = "/home/kemal/Downloads/program deteksi plat
dan wajah/model/deteksiPlat.weights"
cfg = "/home/kemal/Downloads/program deteksi plat
dan wajah/model/deteksiPlat.cfg"
names = "/home/kemal/Downloads/program deteksi plat
dan wajah/model/deteksiplat.names"
pathImage = "/home/kemal/Downloads/program deteksi
plat dan wajah/images/dataparkir/driver1_45deg.jpg"
images = cv2.imread(pathImage)
levelConf = 0.5

## menggunakan fungsi YOLO yang telah dibuat
hasilDeteksi = yolo(weight, cfg, names, images,
levelConf)


## ambil gambar yang di crop
img = hasilDeteksi[0]

## olah gambar
hasilPreprocessing = preprocessing(img)

## disimpan
cv2.imwrite('cropped.jpg', hasilPreprocessing[0])
```

```
### Display the cropped image
# cv2.imshow("cropped", hasilDeteksi[0])
# cv2.imshow('image',hasilDeteksi[1])

# PENGENALAN KARAKTER

## memasukkan parameter
weight2 = "/home/kemal/Downloads/program deteksi
plat dan wajah/model/pengenalanPlat.weights"
cfg2 = "/home/kemal/Downloads/program deteksi plat
dan wajah/model/pengenalan.cfg"
names2 = "/home/kemal/Downloads/program deteksi plat
dan wajah/model/pengenalan.names"

## memasukkan gambar crop yang sudah di save
#images2 = cv2.imread("/home/kemal/Downloads/program
deteksi plat dan wajah/cropped.jpg")
images2 =
cv2.cvtColor(hasilPreprocessing[0],cv2.COLOR_GRAY2RG
B)
#cv2.imshow("cropped", images2)
#cv2.waitKey(0)
#cv2.destroyAllWindows()
## proses pengenalan gamabr menggunakan fungsi YOLO
hasilPengenalan = yolo(weight2, cfg2, names2,
images2, levelConf)
# img2 = hasilPengenalan[1]
print(hasilPengenalan[2])

#simpan data plat nomor
plate= str(hasilPengenalan[2])
```

```
#SIMPAN PLAT NOMOR KE DALAM DATABASE
insertPlat(plate)

#BACA DATA YG ADA DIDATABASE APAKAH UDAH ADA ATAU
BELOM?
status = readData(plate)

#jika data masih 0 maka akan di insert jika tidak
maka sudah ada
if status == 0:
    #encoding wajah
    hasilEncode = findEncoding(images)

    #simpan ke database
    insertDatabase(plate, hasilEncode)
else:
    #jika status = 1
    print("data sudah pernah terecord")

    toc = timeit.default_timer()
    timetotal = toc - tic
    print("waktu ekseskusi adalah",timetotal)

    sys.exit()

#print(hasilEncode)

#simpan ke CSV
#simpanDataPintuMasuk(plate,hasilEncode)


toc = timeit.default_timer()
timetotal = toc - tic
print("waktu ekseskusi adalah",timetotal)
```

```
## menampilkan gambar
# cv2.imshow('pengenalan', hasilPengenalan[1])
# cv2.waitKey(0)
# cv2.destroyAllWindows()
```

3. Exit gate

Kode utama pada gerbang masuk yang akan memanggil beberapa fungsi untuk menjalankan program seperti fungsi deteksi plat, pengenalan karakter, pengenalan wajah, dan query data

```
from yolobykemal import yolo
from imgpreprocesing import preprocessing
from encodingWajah import findEncoding,compareImages
from gateRegister import simpanDataPintuMasuk
import cv2
from queryData import insertDatabase, insertPlat,
exitPlate, insertTimeOut
import sys
import face_recognition as fr
import ast
import pickle
#DETEKSI PLAT

## memasukkan parameter
weight = "/home/kemal/Downloads/program deteksi plat
dan wajah/model/deteksiPlat.weights"
cfg = "/home/kemal/Downloads/program deteksi plat
dan wajah/model/deteksiPlat.cfg"
names = "/home/kemal/Downloads/program deteksi plat
dan wajah/model/deteksiplat.names"
pathImage = "/home/kemal/Downloads/program deteksi
plat dan
wajah/images/dataparkir/driver1_helm_90deg.jpg"
```

```
images = cv2.imread(pathImage)
levelConf = 0.5

## menggunakan fungsi YOLO yang telah dibuat
hasilDeteksi = yolo(weight, cfg, names, images,
levelConf)

## ambil gambar yang di crop
img = hasilDeteksi[0]

## olah gambar
hasilPreprocessing = preprocessing(img)

## disimpan
cv2.imwrite('cropped.jpg', hasilPreprocessing[0])

### Display the cropped image
# cv2.imshow("cropped", hasilDeteksi[0])
# cv2.imshow('image',hasilDeteksi[1])

# PENGENALAN KARAKTER

## memasukkan parameter
weight2 = "/home/kemal/Downloads/program deteksi
plat dan wajah/model/pengenalanPlat.weights"
cfg2 = "/home/kemal/Downloads/program deteksi plat
dan wajah/model/pengenalan.cfg"
names2 = "/home/kemal/Downloads/program deteksi plat
dan wajah/model/pengenalan.names"

## memasukkan gambar crop yang sudah di save
images2 = cv2.imread("/home/kemal/Downloads/program
deteksi plat dan wajah/cropped.jpg")
```

```
## proses pengenalan gamabr menggunakan fungsi YOLO
hasilPengenalan = yolo(weight2, cfg2, names2,
images2, levelConf)
# img2 = hasilPengenalan[1]
# print(hasilPengenalan[2])

#simpan data plat nomor
plate= str(hasilPengenalan[2])

#cari plat nomor di database dan ambil data wajah

platnWajah = exitPlate(plate)
print(platnWajah)
if not platnWajah:
        print("Data tidak ditemukan")
        sys.exit()

for i in range(len(platnWajah)):
    idParkir, wajahPickled = platnWajah[i]

print(idParkir)

face_data = pickle.loads(wajahPickled)

print("ini merupakan data wajah yg lama:",face_data)
print(type(face_data))

#lihat wajah baru
hasilEncode = findEncoding(images)
print(hasilEncode)
#kalo bisa di encode ariel yang [] itu gausah dalam
bentuk satuan, list semuanya biar enak copy pastenya
```

```
results =
fr.compare_faces([face_data],hasilEncode,tolerance=0
.7)
faceDis = fr.face_distance([face_data],hasilEncode)

if results[0] == True:
    print("Wajah cocok gerbang terbuka!")
    insertTimeOut(idParkir)
else:
    print("Wajah tidak cocok")
print(results,faceDis)

# #jika data masih 0 maka akan di insert jika tidak
maka sudah ada
# if isiPlat == 0:
#     print("data TIDAK DITEMUKAN di database")
#     sys.exit()
# else:
```

4. GST Streamer

Akses kamera pada jetson nano menggunakan GST Streamer

```
# gstreamer_pipeline returns a GStreamer pipeline
for capturing from the CSI camera
# Defaults to 1920x1080 @ 30fps
# Flip the image by setting the flip_method (most
common values: 0 and 2)
# display_width and display_height determine the
size of the window on the screen
# Notice that we drop frames if we fall outside the
processing time in the appsink element
```

```
import cv2

def gstreamer_pipeline(
    capture_width=1920,
    capture_height=1080,
    display_width=960,
    display_height=540,
    framerate=30,
    flip_method=2,
):
    return (
        "nvarguscamerasrc ! "
        "video/x-raw(memory:NVMM), "
        "width=(int)%d, height=(int)%d,
framerate=(fraction)%d/1 ! "
        "nvvidconv flip-method=%d ! "
        "video/x-raw, width=(int)%d, height=(int)%d,
format=(string)BGRx ! "
        "videoconvert ! "
        "video/x-raw, format=(string)BGR ! appsink
drop=True"
        % (
            capture_width,
            capture_height,
            framerate,
            flip_method,
            display_width,
            display_height,
        )
    )
```

5. Query Data

CRUD data pada entrance gate dan exit gate.

```
import mysql.connector
from datetime import datetime
import pickle


# INSERT PLAT NOMOR
def insertPlat(plat):
    con = mysql.connector.connect(
        host = "localhost",
        user = "root",
        password = "kemal123",
        db = 'dbparkir'
        )
    #if con.is_connected():
        #print("Koneksi ke dbparkir bagian INSERT
PLAT 1 Berhasil")

    tabel = con.cursor()
    sql = "INSERT IGNORE INTO LicensePlate
(platNomor) VALUES (%s)"
    data = (plat,)
    tabel.execute(sql,data)

    con.commit()
    #print("data plat nomor berhasil disimpan")


def insertDatabase(plat,wajah):
    #insert DATA

    con = mysql.connector.connect(
```

```
        host = "localhost",
        user = "root",
        password = "kemal123",
        db = 'dbparkir',
        )

    #if con.is_connected():
    #    print("Koneksi ke dbparkir bagian INSERT
DATA SEMUANYA Berhasil")

    ## Pickle the list into a string
    face_pickled_data = pickle.dumps(wajah)

    tabel = con.cursor()
    sql = "INSERT INTO
ParkingInformation(encodeWajah , platID , waktuMasuk
, status) VALUES(%s,(SELECT id FROM LicensePlate
WHERE platNomor = %s),%s, %s);"
    now = datetime.now().strftime('%Y-%m-%d
%H:%M:%S')
    data = (face_pickled_data,plat,now,1)
    tabel.execute(sql,data)
    con.commit()

    print("data record berhasil disimpan")

def readData(plat):
    #READ DATA ON ENTRANCE
    con = mysql.connector.connect(
        host = "localhost",
        user = "root",
        password = "kemal123",
        db = 'dbparkir'
        )
```

```
    #if con.is_connected():
    #    print("Koneksi ke dbparkir bagian READ DATA
Berhasil")

    tabel = con.cursor()
    # tabel.execute("SELECT platID FROM
ParkingInformation WHERE platID=1 AND status <> 1")
    sql = "SELECT platID FROM ParkingInformation
WHERE platID=(SELECT id FROM LicensePlate WHERE
platNomor = (%s)) AND status = 1"
    data = (plat,)
    tabel.execute(sql,data)
    myresult = tabel.fetchall()
    y = len(myresult)
    return y

def exitPlate(plat):
    #READ DATA ON ENTRANCE
    con = mysql.connector.connect(
        host = "localhost",
        user = "root",
        password = "kemal123",
        db = 'dbparkir'
        )

    if con.is_connected():
        print("Koneksi ke dbparkir bagian READ DATA
Berhasil")

    tabel = con.cursor()
    # tabel.execute("SELECT platID FROM
ParkingInformation WHERE platID=1 AND status <> 1")
```

```
    sql = "SELECT idParkir, encodeWajah FROM
ParkingInformation WHERE platID=(SELECT id FROM
LicensePlate WHERE platNomor = (%s)) AND status = 1"
    data = (plat,)
    tabel.execute(sql,data)

    myresult = tabel.fetchall()

    con.commit()
    return myresult

def insertTimeOut(idParkir):
    #READ DATA ON ENTRANCE
    con = mysql.connector.connect(
        host = "localhost",
        user = "root",
        password = "kemal123",
        db = 'dbparkir'
        )

    if con.is_connected():
        print("Koneksi ke dbparkir bagian READ DATA
Berhasil")

    tabel = con.cursor()
    # tabel.execute("SELECT platID FROM
ParkingInformation WHERE platID=1 AND status <> 1")
    sql = "UPDATE ParkingInformation SET waktuKeluar
= %s, status = 0 WHERE idParkir = %s AND status = 1"
    now = datetime.now().strftime('%Y-%m-%d
%H:%M:%S')
    data = (now,idParkir)
    tabel.execute(sql,data)
    con.commit()
```

6. YOLO

Algoritma YOLO

```
def yolo(weight, cfg, names, img, levelConf=0.5):
    #import packages
    import enum
    import cv2
    import numpy as np
    import sys
    level_confidence = levelConf
    #inisialisasi array
    class_ids = []
    confidences = []
    boxes = []
    texts = []
    cropped = []
    x_array = []
    font = cv2.FONT_HERSHEY_PLAIN

    net = cv2.dnn.readNet(weight,cfg)

    #load YOLO network
    classes = []
    with open(names,"r") as f:
        classes = [line.strip() for line in
f.readlines()]

    #dapatkkan output layer
    layer_names = net.getLayerNames()
    output_layers = [layer_names[i - 1] for i in
net.getUnconnectedOutLayers()]
```

```
    #uncoment kalao mau di resize
    #img = cv2.resize(img, None, fx = 0.2, fy = 0.2)
    height, width, channels = img.shape

    #detecting object
    blob = cv2.dnn.blobFromImage(img, 0.00392,
(416,416),(0,0,0),True,crop = False)

    # proses gambar
    net.setInput(blob)
    outs = net.forward(output_layers)

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

                #koordinat kotak
                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)

    #kasih warna untuk setiap kelas
    colors =
np.random.uniform(0,255,size=(len(boxes),3))
    #paksa untuk menampilkan hanya 1 box
    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(classes[class_ids[i]])
            texts.append([x,label])
            #kasih warna
            color = colors[i]
            cv2.rectangle(img,(x,y),(x+w,y+h),color,
2)
            #cv2.putText(img,label,(x,y +
30),font,2,color,3)
            cropped_image = img[y:y+h,x:x+w] #
Slicing to crop the image
    texts.sort()

    textsJoin =""

    for i in range (len(texts)):
        textsJoin = textsJoin + texts[i][1]

    if not textsJoin:
        print("gambar tidak terdeteksi!")
        sys.exit()
    else:
```

```
        return [cropped_image, img, textsJoin]
```

7. Training untuk mendapatkan file bobot menggunakan google colabs

```
# change makefile to have GPU and OPENCV enabled
%cd darknet
!sed -i 's/OPENCV=0/OPENCV=1/' Makefile
!sed -i 's/GPU=0/GPU=1/' Makefile
!sed -i 's/CUDNN=0/CUDNN=1/' Makefile
!sed -i 's/CUDNN_HALF=0/CUDNN_HALF=1/' Makefile

# verify CUDA
!/usr/local/cuda/bin/nvcc --version

# make darknet (builds darknet so that you can then use the darknet exec
utable file to run or train object detectors)
!make

# define helper functions
def imShow(path):
  import cv2
  import matplotlib.pyplot as plt
  %matplotlib inline

  image = cv2.imread(path)
  height, width = image.shape[:2]
  resized_image = cv2.resize(image,(3*width, 3*height), interpolation =
cv2.INTER_CUBIC)

  fig = plt.gcf()
  fig.set_size_inches(18, 10)
  plt.axis("off")
  plt.imshow(cv2.cvtColor(resized_image, cv2.COLOR_BGR2RGB))
  plt.show()

# use this to upload files
def upload():
  from google.colab import files
  uploaded = files.upload()
```

```
  for name, data in uploaded.items():
    with open(name, 'wb') as f:
      f.write(data)
      print ('saved file', name)

# use this to download a file
def download(path):
  from google.colab import files
  files.download(path)
```

#Hubungkan dengan Google Drive

```
%cd ..
from google.colab import drive
drive.mount('/content/gdrive')
# this creates a symbolic link so that now the path /content/gdrive/My\
Drive/ is equal to /mydrive
!ln -s /content/gdrive/My\ Drive/ /mydrive
!ls /mydrive

# cd back into the darknet folder to run detections
%cd darknet
```

Ambil dataset

```
# this is where my datasets are stored within my Google Drive (I created
 a yolov4 folder to store all important files for custom training)
!ls /mydrive/Colab\ Notebooks/Colab_kemal/plate_recognition
# copy over both datasets into the root directory of the Colab VM (comme
nt out test.zip if you are not using a validation dataset)
!cp /mydrive/Colab\ Notebooks/Colab_kemal/plate_recognition/obj.zip ../
!cp /mydrive/Colab\ Notebooks/Colab_kemal/plate_recognition/test.zip ../
# unzip the datasets and their contents so that they are now in /darknet
/data/ folder
!unzip ../obj.zip -d data/
!unzip ../test.zip -d data/
```

#Upload CFG file

```
# upload the custom .cfg back to cloud VM from Google Drive
!cp /mydrive/Colab\ Notebooks/Colab_kemal/plate_recognition/yolov4-
obj.cfg ./cfg
```

```
# upload the obj.names and obj.data files to cloud VM from Google Drive
!cp /mydrive/Colab\ Notebooks/Colab_kemal/plate_recognition/obj.names ./
data
```

```
!cp /mydrive/Colab\ Notebooks/Colab_kemal/plate_recognition/obj.data  ./
data
```

```
# upload the generate_train.py and generate_test.py script to cloud VM f
rom Google Drive
!cp /mydrive/Colab\ Notebooks/Colab_kemal/plate_recognition/generate_tra
in.py ./
!cp /mydrive/Colab\ Notebooks/Colab_kemal/plate_recognition/generate_tes
t.py ./
!python generate_train.py
!python generate_test.py
# verify that the newly generated train.txt and test.txt can be seen in
our darknet/data folder
!ls data/
```

```
!wget https://github.com/AlexeyAB/darknet/releases/download/darknet_yolo
_v3_optimal/yolov4.conv.137
```

```
#training

# train your custom detector! (uncomment %%capture below if you run into
 memory issues or your Colab is crashing)
# %%capture
!./darknet detector train data/obj.data cfg/yolov4-
obj.cfg yolov4.conv.137 -dont_show -map
```