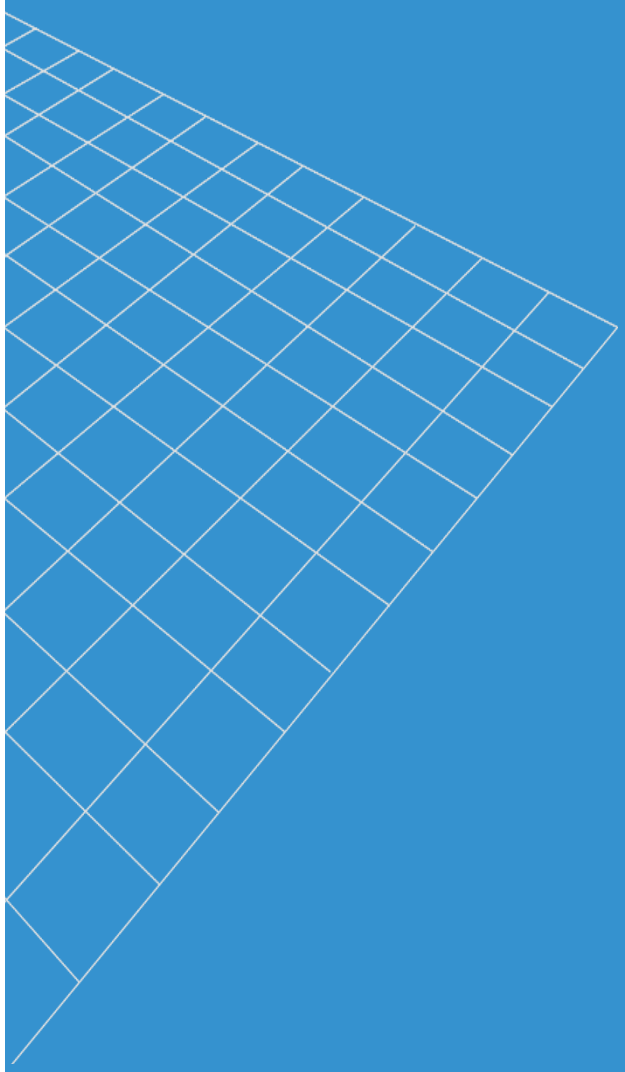
[PANDUAN APLIKASI] MANAJEMEN LAMPU LALU LINTAS BERBASIS PENGOLAHAN CITRA DIGITAL DAN KECERDASAN BUATAN



DECEMBER 62**020**

Ary Setijadi P.
Rahadiyan Yusuf
Reza Darmakusuma
Ach Maulana Habibi Y.

MANAGEMEN LAMPU LALU LINTAS
BERBASIS KECERDASAN BUATAN

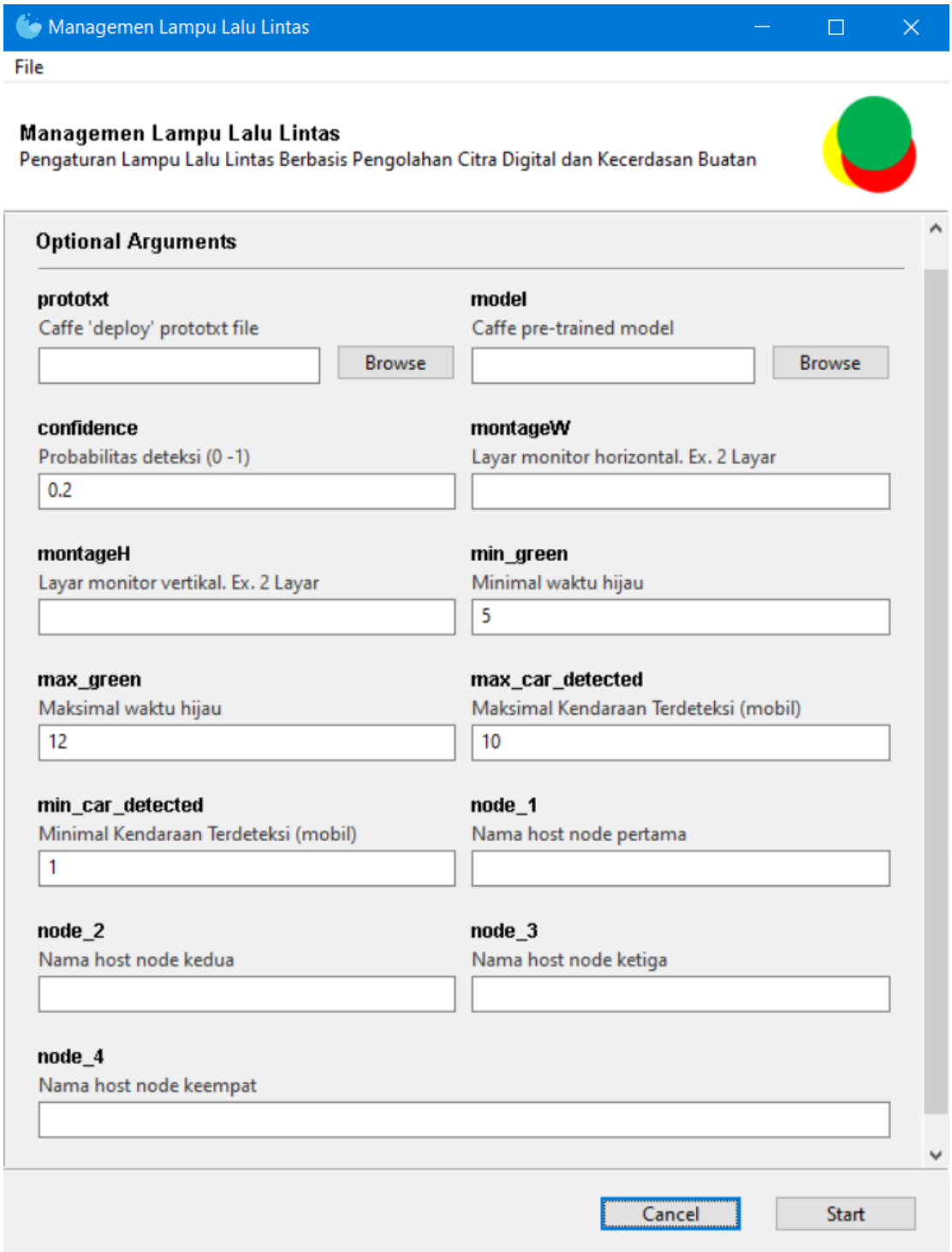
Panduan Aplikasi

Draft ini menjelaskan tentang panduan mengatur parameter pada aplikasi manajemen lalu lintas berbasi pengolahan citra dan kecerdasan buatan. Fitur GUI (*Graphical User Interface*) dibuat untuk memudahkan pengguna untuk mengatur berbagai hal yang diperlukan agar aplikasi berjalan dengan normal. GUI dibuat dengan mengubah argparser pada program menjadi Gooeyparse.

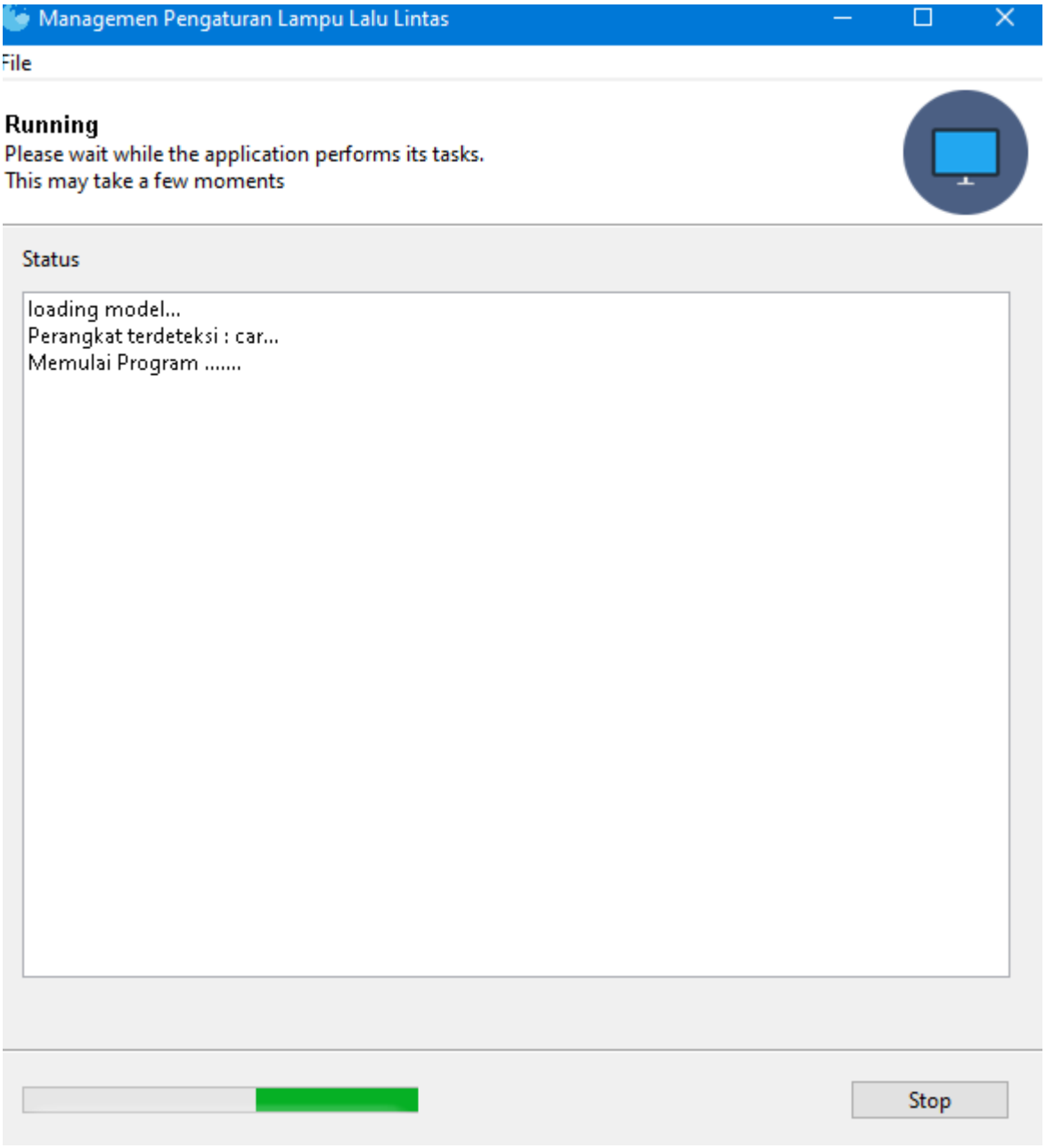
Untuk memulai aplikasi, admin pada pengendali pusat (windows user) membuka command promt / windows powershell. Lalu masuk directory file aplikasi berada. Program dimulai dengan mengetikkan perintah sebagai berikut:

```
# python3 main.py
```

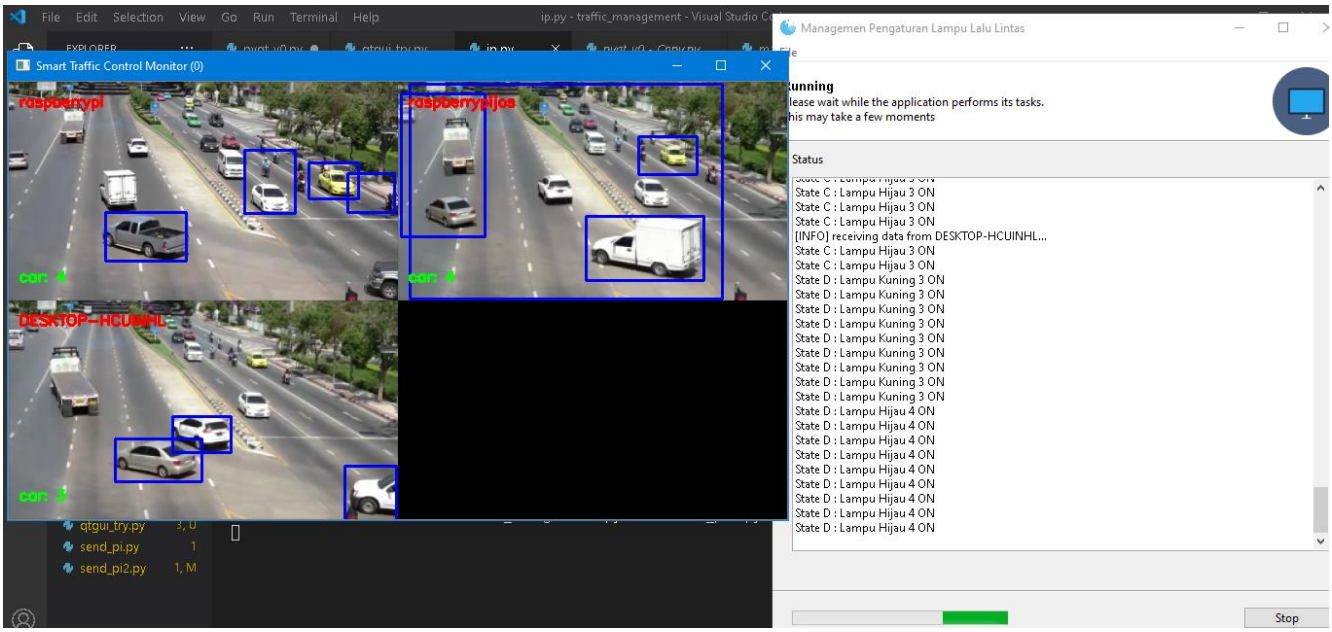
Berikut adalah tampilan awal ketika program main.py dieksekusi. Pengguna diharuskan untuk mengisi kolom dan menginput file yang diperlukan untuk memulai program.

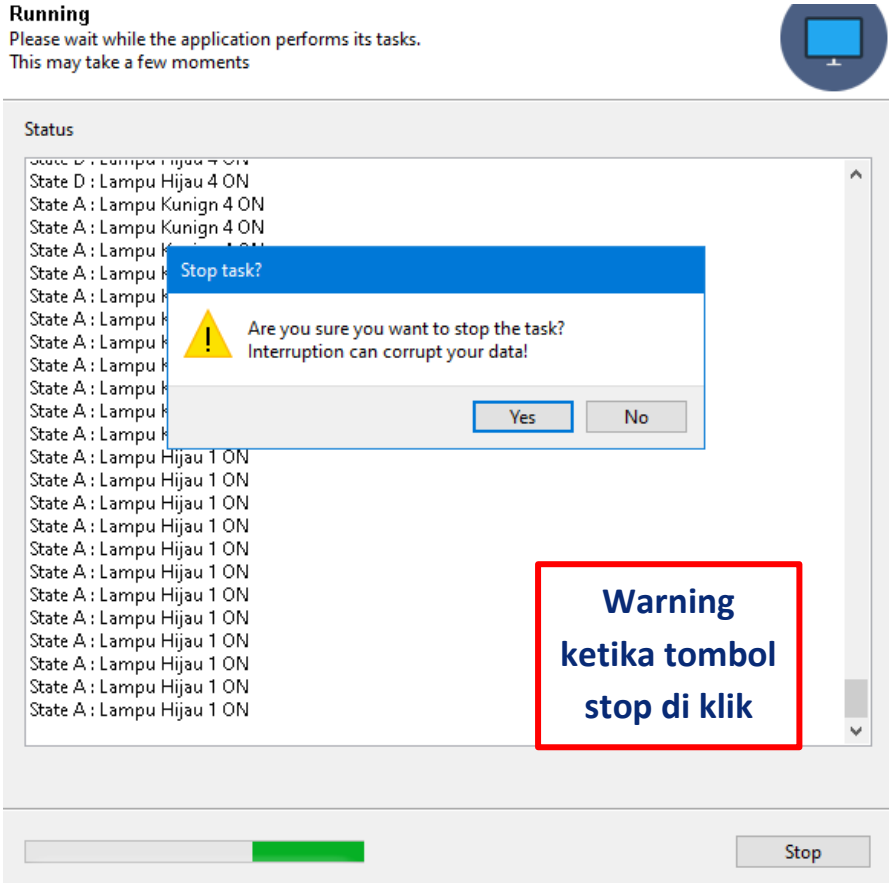


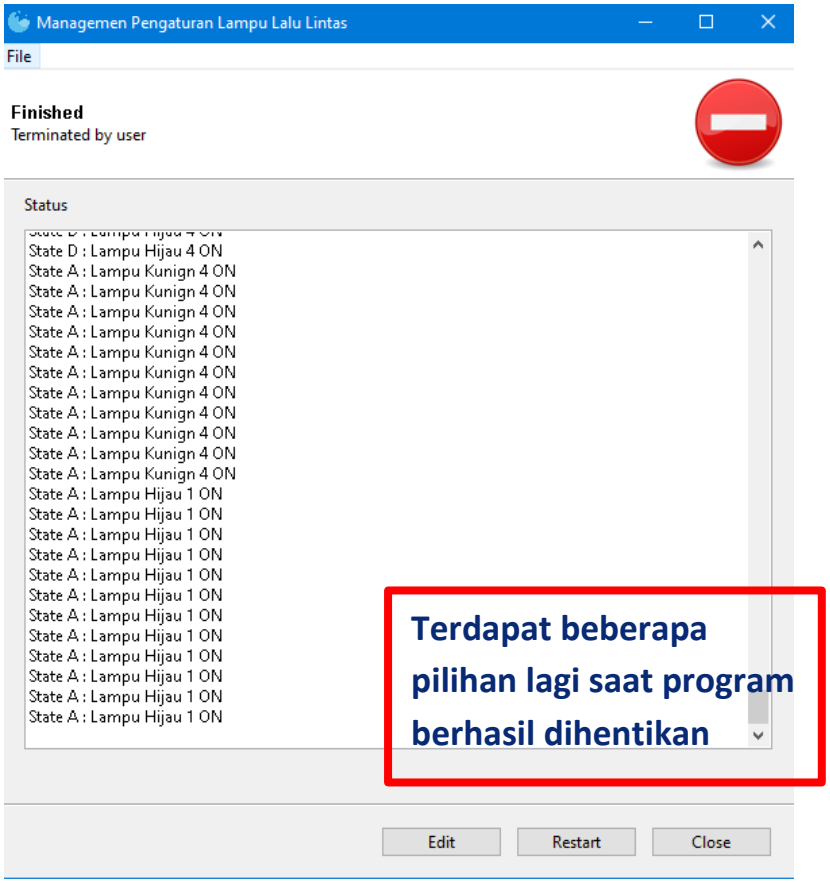
Ketika parameter sudah dimasukkan, maka user memencet tombol Start. Lalu program akan memulai proses. Terdapat tombol Stop untuk menghentikan program.

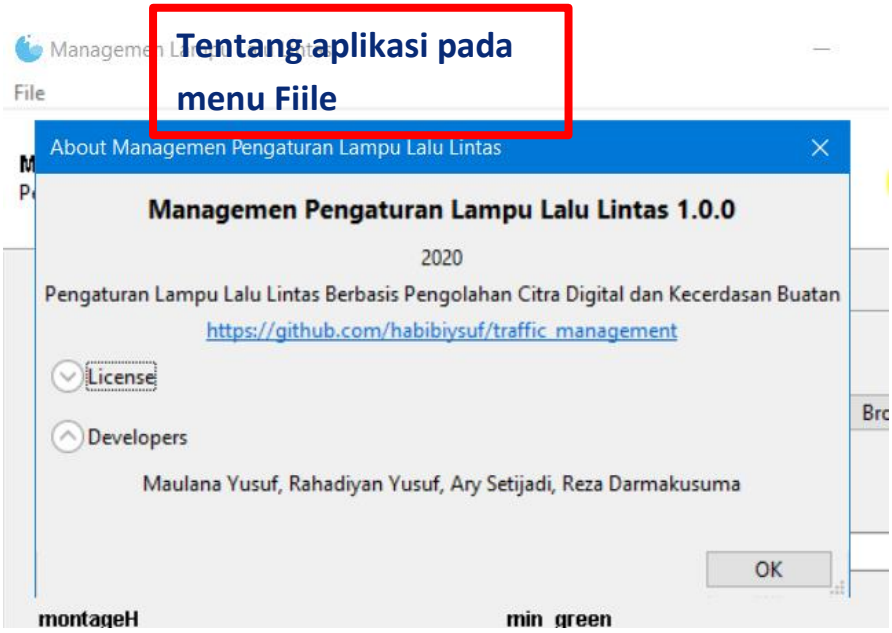


Tampilan ketika program menerima input dan menghasilkan state lampu lalu lintas. Jendela visualisasi output akan muncul secara pop-up pada jendela baru.






Managemen Pengaturan Lampu Lalu Lintas
File
Finished
Terminated by user
Status
State D : Lampu Hijau 4 ON
State A : Lampu Kunign 4 ON
State A : Lampu Kunign 4 ON
State A : Lampu Kunign 4 ON
State A : Lampu Kunign 4 ON
State A : Lampu Kunign 4 ON
State A : Lampu Kunign 4 ON
State A : Lampu Kunign 4 ON
State A : Lampu Kunign 4 ON
State A : Lampu Kunign 4 ON
State A : Lampu Kunign 4 ON
State A : Lampu Kunign 4 ON
State A : Lampu Hijau 1 ON
State A : Lampu Hijau 1 ON
State A : Lampu Hijau 1 ON
State A : Lampu Hijau 1 ON
State A : Lampu Hijau 1 ON
State A : Lampu Hijau 1 ON
State A : Lampu Hijau 1 ON
State A : Lampu Hijau 1 ON
State A : Lampu Hijau 1 ON
State A : Lampu Hijau 1 ON
State A : Lampu Hijau 1 ON
Terdapat beberapa pilihan lagi saat program berhasil dihentikan
Edit
Restart
Close


Tentang aplikasi pada menu Fiile
File
About Managemen Pengaturan Lampu Lalu Lintas
Managemen Pengaturan Lampu Lalu Lintas 1.0.0
2020
Pengaturan Lampu Lalu Lintas Berbasis Pengolahan Citra Digital dan Kecerdasan Buatan
https://github.com/habibiysuf/traffic_management
License
Developers
Maulana Yusuf, Rahadiyan Yusuf, Ary Setijadi, Reza Darmakusuma
OK
montageH
min_green

SOURCE_CODE

```
from imutils import build_montages
from datetime import datetime
import numpy as np
import imagezmq
import argparse
import imutils
import cv2
import random
import time
import threading
from threading import Thread
from gooey import Gooey, GooeyParser

@Gooey(
    program_name="Managemen Lampu Lalu Lintas",
    program_description="Pengaturan Lampu Lalu Lintas Berbasis Pengolahan Citra Digita
l dan Kecerdasan Buatan",
    image_dir = "C:/Users/achma/Documents/traffic_management",
    menu=[{'name': 'File', 'items': [{
        'type': 'AboutDialog',
        'menuTitle': 'About',
        'name': 'Managemen Pengaturan Lampu Lalu Lintas',
        'description': 'Pengaturan Lampu Lalu Lintas Berbasis Pengolahan Citra Digital
 dan Kecerdasan Buatan',
        'version': '1.0.0',
        'copyright': '2020',
        'website': 'https://github.com/habibiysuf/traffic_management',
        'developer': "Maulana Yusuf, Rahadiyan Yusuf, Ary Setijadi, Reza Darmakusuma",
        'license': 'TMDG ITB' }]
        }]

)
def main():
    parser = GooeyParser(description="Pengaturan Parameter Sistem")
    #ap = argparse.ArgumentParser()
    parser.add_argument("-p", "--prototxt", required=True,
        help="Caffe 'deploy' prototxt file", widget='FileChooser')
    parser.add_argument("-m", "--model", required=True,
        help="Caffe pre-trained model", widget='FileChooser')
    parser.add_argument("-c", "--confidence", type=float, default=0.2,
        help="Probabilitas deteksi (0 -1)")
    parser.add_argument("-mW", "--montageW", required=True, type=int,
        help="Layar monitor horizontal. Ex. 2 Layar")
```

```
    parser.add_argument("-mH", "--montageH", required=True, type=int,
        help="Layar monitor vertikal. Ex. 2 Layar ")
    parser.add_argument("-mig", "--
min_green", help="Minimal waktu hijau", type=int, default=5)
    parser.add_argument("-mag", "--
max_green", help="Maksimal waktu hijau",type=int, default=12)
    parser.add_argument("-macd", "--
max_car_detected", help = "Maksimal Kendaraan Terdeteksi (mobil)", type=int, default=1
0 )
    parser.add_argument("-micd", "--
min_car_detected",help = "Minimal Kendaraan Terdeteksi (mobil)", type=int, default=1)
    parser.add_argument("-sc1", "--
node_1", help = "Nama host node pertama", required=True)
    parser.add_argument("-sc2", "--
node_2", help = "Nama host node kedua", required=True)
    parser.add_argument("-sc3", "--
node_3", help = "Nama host node ketiga", required=True)
    parser.add_argument("-sc4", "--
node_4", help = "Nama host node keempat", required=True)
    args = vars(parser.parse_args())

    max_car_detect = args["max_car_detected"]
    min_car_detect = args["min_car_detected"]
    min_green = args["min_green"]
    max_green = args["max_green"]

    open_akhir = True
    open_awal = True

    state_a = True
    state_b = False
    state_c = False
    state_d = False

    imageHub = imagezmq.ImageHub()
    CLASSES = ["background", "aeroplane", "bicycle", "bird", "boat",
        "bottle", "bus", "car", "cat", "chair", "cow", "diningtable",
        "dog", "horse", "motorbike", "person", "pottedplant", "sheep",
        "sofa", "train", "tvmonitor"]

    print("loading model...")
    net = cv2.dnn.readNetFromCaffe(args["prototxt"], args["model"])

    CONSIDER = set(["car"])
    objCount = {obj: 0 for obj in CONSIDER}
    frameDict = {}
```

```
lastActive = {}
lastActiveCheck = datetime.now()

ESTIMATED_NUM_PIS = 4
ACTIVE_CHECK_PERIOD = 10
ACTIVE_CHECK_SECONDS = ESTIMATED_NUM_PIS * ACTIVE_CHECK_PERIOD

mW = args["montageW"]
mH = args["montageH"]
print("Perangkat terdeteksi : {}...".format(", ".join(obj for obj in
    CONSIDER)))

gate = False
state_1 = True
devices = []

print("Memulai Program .......")
while True:

    (rpiName, frame) = imageHub.recv_image()
    imageHub.send_reply(b'OK')
    if rpiName not in lastActive.keys():
        print("[INFO] receiving data from {}...".format(rpiName))
        devices.append(rpiName)

    lastActive[rpiName] = datetime.now()
    frame = imutils.resize(frame, width=400)
    (h, w) = frame.shape[:2]


    blob = cv2.dnn.blobFromImage(cv2.resize(frame, (300, 300)),
        0.007843, (300, 300), 127.5)

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

    objCount = {obj: 0 for obj in CONSIDER}
    for i in np.arange(0, detections.shape[2]):
        confidence = detections[0, 0, i, 2]

        if confidence > args["confidence"]:

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

            if CLASSES[idx] in CONSIDER:
```

```
                    objCount[CLASSES[idx]] += 1

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

                    cv2.rectangle(frame, (startX, startY), (endX, endY),
                        (255, 0, 0), 2)
        cv2.putText(frame, rpiName, (10, 25),cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255
), 2)

        label = ", ".join("{}: {}".format(obj, count) for (obj, count) in objCount.ite
ms())
        cv2.putText(frame, label, (10, h - 20), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255
,0), 2)

        if len(devices) == 1:
            count_a = objCount["car"]
            count_b = random.randint(1,9)
            count_c = random.randint(1,6)
            count_d = random.randint(1,7)
        elif len(devices) == 2:
            if devices[0] == args["node_1"]:
                count_a = objCount["car"]
            else:
                count_a = random.randint(1,9)
            if devices[1] == args["node_2"]:
                count_b = objCount["car"]
            else:
                count_b = random.randint(1,9)
            count_c = random.randint(1,6)
            count_d = random.randint(1,7)

        elif len(devices) == 3:
            if devices[0] == args["node_1"]:
                count_a = objCount["car"]
            else:
                count_a = random.randint(1,9)
            if devices[1] == args["node_2"]:
                count_b = objCount["car"]
            else:
                count_b = random.randint(1,9)
            if devices[2] == args["node_3"]:
                count_b = objCount["car"]
            else:
                count_b = random.randint(1,9)
```

```
            count_d = random.randint(1,7)

        elif len(devices) == 4:
            if devices[0] == args["node_1"]:
                count_a = objCount["car"]
            else:
                count_a = random.randint(1,9)
            if devices[1] == args["node_2"]:
                count_b = objCount["car"]
            else:
                count_b = random.randint(1,9)
            if devices[2] == args["node_3"]:
                count_b = objCount["car"]
            else:
                count_b = random.randint(1,9)
            if devices[3] == args["node_4"]:
                count_b = objCount["car"]
            else:
                count_b = random.randint(1,9)

        ###################### 1
        if (count_a <= min_car_detect and count_a != 0):
            output_1 = min_green
        elif (count_a > min_car_detect and count_a <= max_car_detect):
            t_output_1 = (max_car_detect - count_a) / (max_car_detect - min_car_detect
)
            output_1 = max_green - t_output_1*(max_car_detect - min_car_detect)
        elif (count_a == 0):
            output_1 = 0
        elif (count_a > max_car_detect):
            output_1 = max_green
        ###################### 2
        if (count_b <= min_car_detect and count_b != 0):
            output_2 = min_green
        elif (count_b > min_car_detect and count_b <= max_car_detect):
            t_output_2 = (max_car_detect - count_b) / (max_car_detect - min_car_detect
)
            output_2 = max_green - t_output_2*(max_car_detect - min_car_detect)
        elif (count_b == 0):
            output_2 = 0
        elif (count_b > max_car_detect):
            output_2 = max_green
        ###################### 3
        if (count_c <= min_car_detect and count_c != 0):
            output_3 = min_green
        elif (count_c > min_car_detect and count_c <= max_car_detect):
```

```
            t_output_3 = (max_car_detect - count_c) / (max_car_detect - min_car_detect
)
            output_3 = max_green - t_output_3*(max_car_detect - min_car_detect)
        elif (count_c == 0):
            output_3 = 0
        elif (count_c > max_car_detect):
            output_3 = max_green
        ################## 4
        if (count_d <= min_car_detect and count_d != 0):
            output_4 = min_green
        elif (count_d > min_car_detect and count_d <= max_car_detect):
            t_output_4 = (max_car_detect - count_d) / (max_car_detect - min_car_detect
)
            output_4 = max_green - t_output_4*(max_car_detect - min_car_detect)
        elif (count_d == 0):
            output_4 = 0
        elif (count_d > max_car_detect):
            output_4 = max_green


        #print("Source 1 : "+str(output_1)+ " Source 2 : "+str(output_2)+ " Source 3 :
 "+str(output_3)+ " Source 4 : "+str(output_4), end="\r")
        #susunan (output_1, output_2, output_3, output_4)
        if (state_a == True):
            if (open_akhir == True):
                TTA = time.time()
                KA = TTA + 3
                task_print = True
            open_akhir = False
            if task_print == True:
                print("State A : Lampu Kunign 4 ON")
            if (time.time() > KA):
                task_print = False
                if (open_awal == True):
                    TA = time.time()
                    HA = TA + output_1
                open_awal = False
                print("State A : Lampu Hijau 1 ON")
                if (time.time() > HA):
                    state_b = True
                    state_a = False
                    open_akhir = True
                    open_awal = True


        elif (state_b == True):
```

```
            if (open_akhir == True):
                TTB = time.time()
                KB = TTB + 3
                task_print = True
            open_akhir = False
            if task_print == True:
                print("State B : Lampu Kuning 1 ON")
            if (time.time() > KB):
                task_print = False
                if (open_awal == True):
                    TB = time.time()
                    HB = TB + output_2
                open_awal = False
                print("State B : Lampu Hijau 2 ON")
                if (time.time() > HB):
                    state_c = True
                    state_b = False
                    open_akhir = True
                    open_awal = True


        elif (state_c == True):
            if (open_akhir == True):
                TTC = time.time()
                KC = TTC + 3
                task_print = True
            open_akhir = False
            if task_print == True:
                print("State C : Lampu Kuning 2 ON")
            if (time.time() > KC):
                task_print = False
                if (open_awal == True):
                    TC = time.time()
                    HC = TC + output_3
                open_awal = False
                print("State C : Lampu Hijau 3 ON")
                if (time.time() > HC):
                    state_d = True
                    state_c = False
                    open_akhir = True
                    open_awal = True

        elif (state_d == True):
            if (open_akhir == True):
                TTD = time.time()
                KD = TTD + 3
```

```
                task_print = True
            open_akhir = False
            if task_print == True:
                print("State D : Lampu Kuning 3 ON")
            if (time.time() > KD):
                task_print = False
                if (open_awal == True):
                    TD = time.time()
                    HD = TD + output_4
                open_awal = False
                print("State D : Lampu Hijau 4 ON")
                if (time.time() > HD):
                    state_a = True
                    state_d = False
                    open_akhir = True
                    open_awal = True

        frameDict[rpiName] = frame
        if (gate == True):
            break
        montages = build_montages(frameDict.values(), (w, h), (mW, mH))

        for (i, montage) in enumerate(montages):
            cv2.imshow("Smart Traffic Control Monitor ({})".format(i),
                montage)

        key = cv2.waitKey(1) & 0xFF
        if (datetime.now() - lastActiveCheck).seconds > ACTIVE_CHECK_SECONDS:

            for (rpiName, ts) in list(lastActive.items()):
                if (datetime.now() - ts).seconds > ACTIVE_CHECK_SECONDS:
                    print("[INFO] lost connection to {}".format(rpiName))
                    lastActive.pop(rpiName)
                    frameDict.pop(rpiName)

            lastActiveCheck = datetime.now()
        #time.sleep(0.5)

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

    cv2.destroyAllWindows()
```

```python
if __name__ == "__main__":
    main()
```