

BAB IV

HASIL DAN PEMBAHASAN

4.1 Perangkat Lunak

Perangkat Lunak yang digunakan dalam pembuatan sistem ini adalah :

- a. Window 10*
- b. Tensorflow*
- c. Python AQ*
- d. Anaconda*
- e. OpenCV*
- f. Virtualenv*
- g. Jupyter Lab*
- h. GPU Nvidia Gforce 930M*

4.2 Perangkat Keras

Perangkat Keras yang digunakan dalam pembuatan sistem ini adalah :

- a. Camera Webcam Logitech C9220 Pro*
- b. Arduino Uno*
- c. Lampu LED (light-emitting diode)*
- d. Kabel Data USB Type A to B Arduino Mega Printer 50 Cm*
- e. Memory RAM 4 GB*
- f. Harddisk 500 G*
- g. Keyboard*
- h. Mouse*

4.3 Implementasi Program

Implementasi program merupakan tahapan penggambaran dari program yang telah dirancang kepada para pengguna yang nantinya akan menggunakan program Desain Simulasi *Pedestrian Detector* Untuk Lampu penyeberangan *zebra cross*. Manfaat dari adanya implementasi program yaitu dapat memberikan peringatan kepada para pengendara bahwa sedang ada seseorang yang menggunakan jalur penyeberangan *zebra cross*, program akan berfungsi jika membaca objek seseorang *person* dan sistem akan mengirim sinyal pada lampu dan akan mengubah kondisi menjadi menyala pada lampu *led (light-emitting diode)* pin13 di *arduino*.

penilaian tersebut akan masuk pada pengembang dari program yang akan melakukan perbaikan terhadap program dengan tujuan agar program lebih baik lagi. Dan adapun hal-hal yang harus diperhatikan dalam melakukan implementasi program untuk Desain Simulasi *Pedestrian Detector* Untuk Lampu penyeberangan *Zebra Cross*. Yaitu :

1. Implementasi Perangkat Lunak, yaitu implementasi terhadap perangkat lunak (*Software*) yang nantinya menjalankan dari sistem aplikasi yang dibangun.
2. Implementasi Perangkat Keras, yaitu perangkat keras (*hardware*) yang digunakan untuk menjalankan dari sistem aplikasi yang dibangun.
3. Implementasi Antarmuka, yang berisi detail dari tampilan-tampilan yang ada pada perangkat lunak aplikasi yang dibangun tersebut yang terdiri dari nama antarmuka atau file yang mewakilinya tersebut.

Dan berikut merupakan hasil implementasi dari program :

4.3.1 Implementasi Tampilan Pengguna (*User*)

Rancangan tampilan pada Pengguna (*User*) merupakan bentuk hasil rancangan menggunakan *Tensorflow* dan *OpenCV* yang diintegrasikan pada *Jupyter Lab* dan Bahasa Pemrograman *Python*, berikut adalah rancangan tampilan pada bagian pengguna (*User*):

a. Implementasi kondisi awal

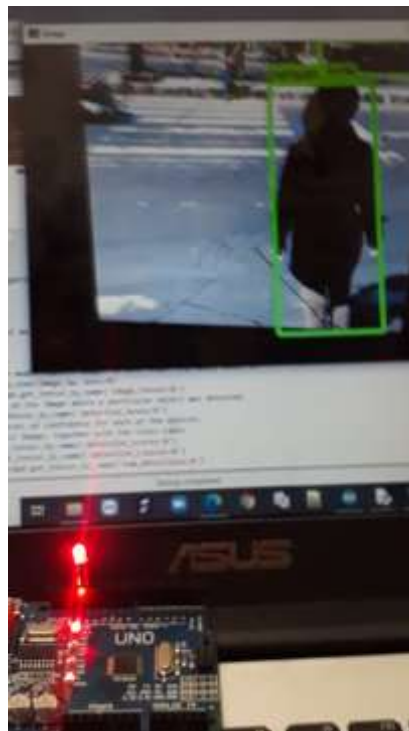
Implementasi kondisi awal merupakan tampilan pertama, Pada Implementasi Kondisi di mana tidak terdapat objek yang melintas pada *webcam* atau pun di jalur lalu lintas. Dan lampu masih dalam kondisi awal berikut adalah tampilan Kondisi awal pada Gambar 4.1 :



Gambar 4.1 Implementasi Kondisi awal Lampu Kondisi awal

b. Implementasi *Object Detection* pada *Pedestrian*

Implementasi *Object detection* pada *pedestrian* untuk mendeteksi *object* yang terdapat di camera *webcam* secara *realtime* pendeteksian *object* pada seseorang “*person*” yang menggunakan jalur lalu lintas penyeberangan *zebra cross* agar sistem mengirimkan perintah kepada *Arduino* untuk menyalakan lampu red pada pin13 dan jika tidak terdapat *object* pada camera maka akan kembali pada kondisi awal yang jika tidak terdapat *object* maka sistem akan mengirimkan perintah ke *Arduino* untuk tidak menghidupkan lampu, berikut adalah tampilan untuk *object detection* pada *object* “*person*” :



Gambar 4.2 Implementasi *Object* yang terdeteksi

4.3.2 Implementasi Tampilan Pengembang (*Developer*)

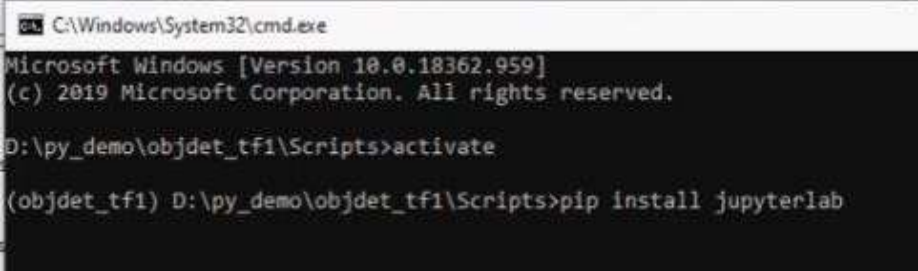
Pada pengimplementasian tampilan pengembang atau *developer* dihasilkan dengan menggunakan *Tensorflow* dan *OpenCV* yang dipergunakan dalam Pendeteksian *Object* yang terjadi di program Desain Simulasi *Pedestrian Detector* untuk Lampu Penyeberangan.

a. Implementasi *Virtualenv*

Implementasi *Virtualenv* pada program ini adalah untuk menghubungkan seluruh aplikasi perangkat lunak secara *virtual enviroment* yaitu tahapan yang dilakukan dengan mengawali tahapan masuk pada aplikasi *tensorflow* dan *OpenCV Virtual environment* adalah sebuah wadah untuk menampung pustaka serta modul dalam suatu proyek pekerjaan agar terisolasi. Ketika kita mengerjakan beberapa aplikasi/proyek dengan modul yang sama akan tetapi membutuhkan versi berbeda, disinilah kita membutuhkan *jupyter*.

1. Pertama *Install Jupyter* lab di command prompt dengan

klik > Pip *Install Jupyter Lab*



```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.18362.959]
(c) 2019 Microsoft Corporation. All rights reserved.

D:\py_demo\objdet_tf1\Scripts>activate

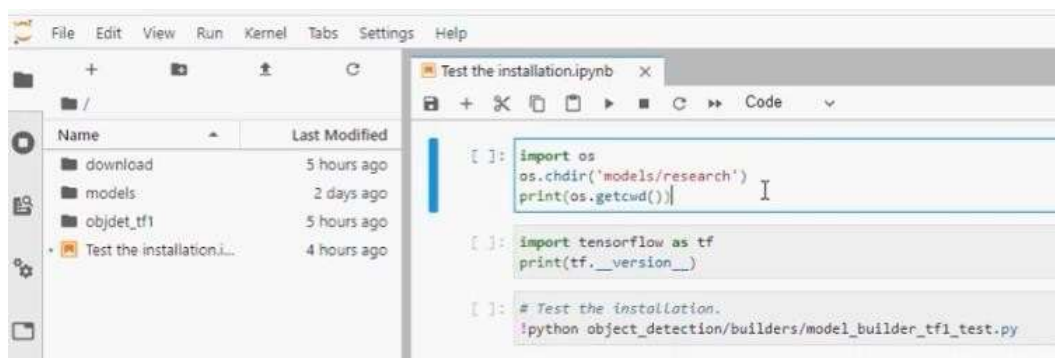
(objdet_tf1) D:\py_demo\objdet_tf1\Scripts>pip install jupyterlab
```

Gambar 4.3 *Install Virtualenv*

```
sqlparse 0.4.1
tensorboard 2.4.1
tensorboard-plugin-profile 2.4.0
tensorboard-plugin-wit 1.8.0
tensorflow-estimator 2.4.0
termcolor 1.1.0
testpath 0.4.4
tornado 6.1
traitlets 5.0.5
unlabeled 1.26.2
virtualenv 20.4.1
wcwidth 0.2.5
webencodings 0.5.1
Werkzeug 1.0.1
wheel 0.36.2
wrapit 1.12.1
zipp 3.4.0

<objdet_tf1> E:\py.demo>
```

Gambar 4.4 Tampilan Setelah *terinstall*



Gambar 4.5 Implementasi *Virtualent* pada Pengembang (*Developed*)

b. Implementasi *Object Detection* pada Gambar

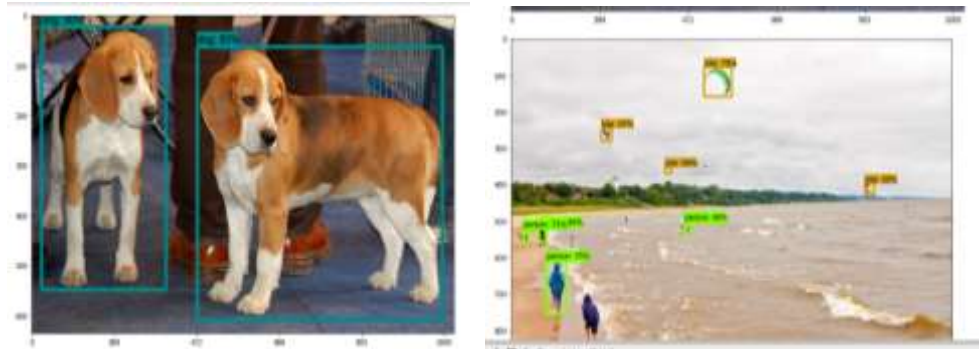
Pengimplementasian pada tahapan ini merupakan pendeteksian objek pada sebuah gambar dimana program akan melakukan pendeteksian dengan menggunakan *Tensorflow* dan *OpenCV* dan di implementasikan di *jupyter lab* dengan menggunakan bahasa *python*. Agar dapat berfungsi Intents untuk melakukan *object detection* yang akan terdeteksi sesuai dengan yang akan di masukan pada sistem.

Berikut tampilan untuk *input* pendeteksian pada gambar, yang terdapat pada *jupyter lab* dengan menggunakan bahasa *python* yang terdapat pada gambar 4.6 :

```
with detection_graph.as_default():
with tf.Session(graph=detection_graph) as sess:
for image_path in TEST_IMAGE_PATHS:
image = Image.open(image_path)
# the array based representation of the image will be used later in order to prepare the
# result image with boxes and labels on it.
image_np = load_image_into_numpy_array(image)
# Expand dimensions since the model expects images to have shape: [1, None, None, 3]
image_np_expanded = np.expand_dims(image_np, axis=0)
image_tensor = detection_graph.get_tensor_by_name('image_tensor:0')
# Each box represents a part of the image where a particular object was detected.
boxes = detection_graph.get_tensor_by_name('detection_boxes:0')
# Each score represent how level of confidence for each of the objects.
# Score is shown on the result image, together with the class label.
scores = detection_graph.get_tensor_by_name('detection_scores:0')
classes = detection_graph.get_tensor_by_name('detection_classes:0')
num_detections = detection_graph.get_tensor_by_name('num_detections:0')
# Actual detection.
(boxes, scores, classes, num_detections) = sess.run(
[boxes, scores, classes, num_detections],
feed_dict={image_tensor: image_np_expanded})
# Visualization of the results of a detection.
vis_util.visualize_boxes_and_labels_on_image_array(
image_np,
np.squeeze(boxes),
np.squeeze(classes).astype(np.int32),
np.squeeze(scores),
category_index,
use_normalized_coordinates=True,
line_thickness=8)
plt.figure(figsize=IMAGE_SIZE)
plt.imshow(image_np)
```

Gambar 4.6 Implementasi *object detection* pada Gambar

Pada implementasi *object detection* pada gambar akan menggunakan program yang dijalankan pada *virtual enviroment* dengan menggunakan bahasa *python* maka proses pendeteksian pada *object* sedang di proses dan akan mengeluarkan *output* sebagai berikut yang terdapat pada gambar 4.7 :



Gambar 4.7 *Output implementasi object detection pada Gambar*

c. Implementasi *object detection realtime* dengan *webcam*

Pengimplementasian *object detection realtime* dengan *webcam* dimana semua data yang akan di deteksi dengan menggunakan camera *webcam* secara langsung, jika ada *object* yang sedang terlintas atau terlihat pada camera *webcam* akan terdeteksi sesuai dengan program yang dirancang dimana *object detection* memiliki 90 *object name* yang terdapat di *library* dan dalam penelitian ini memfokuskan pada seseorang atau "*person*", jadi jika ada seseorang yang sedang menggunakan jalur lalu lintas penyeberangan *zebra cross* akan mendeteksi objek yang terdapat pada gambar 4.8 :


```

[12]: import cv2
      cap = cv2.VideoCapture(0)

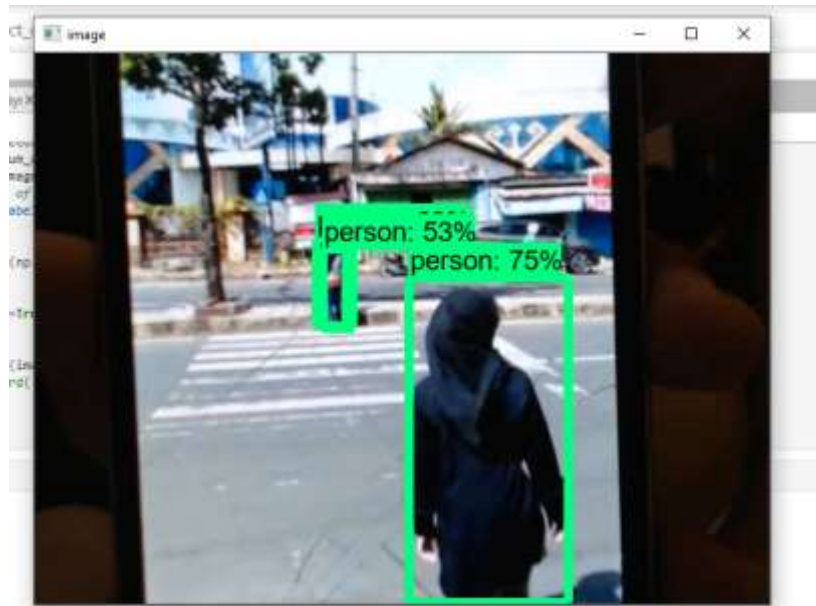
      cap.set(1, 30000)
      width, height = int(cap.get(3)), int(cap.get(4))

      o, f, g = 0, 0, 0
      # Running the tensorflow session
      with detection_graph.as_default():
          with tf.Session(graph=detection_graph) as sess:
              ret = True
              while (ret):
                  ret, image_np = cap.read()
                  # Expand dimensions since the model expects images to have shape: [1, None, None, 3]
                  image_np_expanded = np.expand_dims(image_np, axis=0)
                  image_tensor = detection_graph.get_tensor_by_name('image_tensor:0')
                  # Each box represents a part of the image where a particular object was detected.
                  boxes = detection_graph.get_tensor_by_name('detection_boxes:0')
                  # Each score represent how level of confidence for each of the objects.
                  # Score is shown on the result image, together with the class label.
                  scores = detection_graph.get_tensor_by_name('detection_scores:0')
                  classes = detection_graph.get_tensor_by_name('detection_classes:0')
                  num_detections = detection_graph.get_tensor_by_name('num_detections:0')
                  # Actual detection.
                  (boxes, scores, classes, num_detections) = sess.run(
                      [boxes, scores, classes, num_detections],
                      feed_dict={image_tensor: image_np_expanded})
                  # Visualization of the results of a detection.
                  vis_util.visualize_boxes_and_labels_on_image_array(
                      image_np,
                      np.squeeze(boxes),
                      np.squeeze(classes).astype(np.int32),
                      np.squeeze(scores),
                      category_index,
                      use_normalized_coordinates=True,
                      line_thickness=8)

```

Gambar 4.8 Implementasi *object detection realtime* dengan *webcam*.

Pada gambar 4.9 terdapat hasil dari Implementasi *object detection realtime* dengan *webcam* yang sudah dilakukan secara *realtime* dan bisa juga dengan data dokumentasi yang sudah dikumpulkan berikut hasil *object detection* secara *realtime* dengan *object person* yang dilakukan dengan menggunakan data dokumentasi yang sudah terekam sebelumnya :



Gambar 4.9 Implementasi *object detection realtime* dengan *webcam*.

d. Implementasi *object detection* dan *arduino*

Object detection realtime pengambilan data secara langsung dengan menggunakan *webcam* sebagai data *input* ke program, *pedestrian detector* ini merupakan pendeteksian pada para pengguna jalur penyeberangan *zebra cross* jika ada *object* yang sedang terlintas atau terlihat pada camera *webcam* akan terdeteksi sesuai dengan program yang dirancang dimana *object detection* dalam penelitian ini memfokuskan pada objek “*person*”, jadi jika ada seseorang yang sedang menggunakan jalur lalu lintas penyeberangan *zebra cross* akan mendeteksi dan sistem akan memberikan perintah kepada *arduino* untuk menyalakan lampu *led (light-emitting diode)* pin13 sebagai pertanda sensor sudah terbaca berikut *virtual enviroment* yang di gunakan untuk mendeteksi *object* yang terdapat pada gambar 4.10 :

```
# Visualisasikan hasil deteksi berupa gambar
a, b = vis_util.visualize_boxes_and_labels_on_image_array(
    image_np,
    np.squeeze(boxes),
    np.squeeze(classes).astype(np.int32),
    np.squeeze(scores),
    category_index,
    use_normalized_coordinates=True,
    line_thickness=8)

# Algoritma yang mengirim pesan ke Arduino untuk menhidupkan dan mematikan lampu
if b == 'person':
    f = 0
    if g == 1:
        print('.', end='')
    else:
        a = 1
        print('n: %d' % a, end='')
        if a == 10:
            var = '1'
            ard.write(b'1')
            g = 1
            print('on', end='\n\n')
    elif b != 'person':
        a = 0
        if g == 2:
            print('.', end='')
        else:
            f = 1
            print('f: %d' % f, end='')
            if f == 20:
                var = '0'
                ard.write(b'0')
                g = 2
                print('off', end='\n\n')

cv2.imshow('Object Detection', cv2.resize(image_np, (width, height)))
if cv2.waitKey(25) & 0xFF == ord('q'):
    cv2.destroyAllWindows()
    cap.release()
    break
```

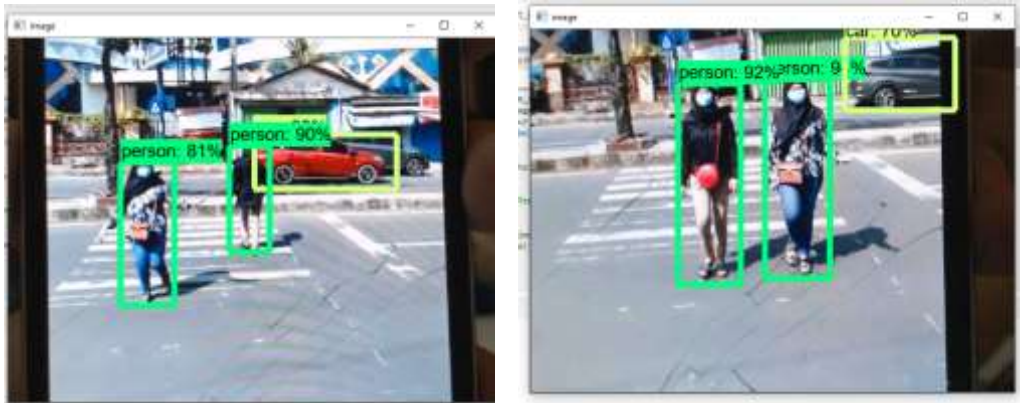
Gambar 4.10 Implementasi *object detection webcam*.

Pada gambar 4.11 tidak ada *object* yang terekam atau *input* ke dalam kamera *webcam* untuk dideteksi karena jika program dalam kondisi ini tidak terjadi pendeteksian dan sistem tetap dalam kondisi awal, tidak ada responsi yang di kirimkan dari *python* ke *arduino* seperti berikut ini :



Gambar4.11 Kondisi tidak ada *object*

Pada gambar 4.13 terjadi proses pendeteksian *object* pada pejalan kaki yang menggunakan jalur penyeberangan *zebra cross* dimana jika ada seseorang “*person*” yang terdeteksi atau *input* kedalam rekaman *camera webcam* secara *realtime* maka sistem akan mengirimkan responsi kepada *Arduino* untuk menyalakan lampu *led (light-emitting diode)* pin13 seperti gambar bawah ini :



Gambar 4.12 *Pedestrian Detector*



Gambar 4.13 *Object terdeteksi pada pedestrian detector*