**Technological Institute of the Philippines**
**Manila**
*Manila 1338 Arlegui Street, Quiapo, Manila*

# Final period:

# Seatwork

*Detection of car/s in a picture*

**Submitted by:**

Ardy N. Ubanos

BSCS-1710590

**Submitted to:**

Engr. Alvin Alon

**Date Submitted:**

October 18, 2018

**Technological Institute of the Philippines**
**Manila**
*Manila 1338 Arlegui Street, Quiapo, Manila*

# Final period:

# Quiz

*Detection of car/s in a video*

**Submitted by:**

Ardy N. Ubanos

BSCS-1710590

**Submitted to:**

Engr. Alvin Alon

**Date Submitted:**

October 18, 2018

**Technological Institute of the Philippines**
**Manila**
*Manila 1338 Arlegui Street, Quiapo, Manila*

# Final period:

# Project

*Detection of cat/s in real-time*

*Video feed*

**Submitted by:**

Ardy N. Ubanos

BSCS-1710590

**Submitted to:**

Engr. Alvin Alon

**Date Submitted:**

October 18, 2018

**SEATWORK:**

```
install.packages("devtools")

install.packages("pkgbuild")

devtools::install_github("bnosac/image", subdir = "image.darknet", build_vignettes = TRUE)

library(pkgbuild)

library(image.darknet)


assignInNamespace("version_info", c(devtools:::version_info, list("3.5" = list(version_min = "3.3.0",
version_max = "99.99.99", path = "bin"))), "devtools")



yolo_tiny_voc <- image_darknet_model(type = "detect",

                                        model = "tiny-yolo-voc.cfg",

                                      weights = system.file(package="image.darknet", "models", "tiny-yolo-voc.weights"),

                                      labels = system.file(package="image.darknet", "include", "darknet", "data", "voc.names"))


x <- image_darknet_detect(file = "E:/dspfinal/sw/car.png",

                        object = yolo_tiny_voc,

                        threshold = 0.5)
```

QUIZ:

```
import cv2

import numpy as np


camera = cv2.VideoCapture ("C:/Python27/video.avi")

car_cascade = cv2.CascadeClassifier('C:/Python27/cars.xml')

while True:

    (grabbed,frame) = camera.read()

    grayvideo = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    cars = car_cascade.detectMultiScale(grayvideo, 1.1, 1)

    for (x,y,w,h) in cars:

     cv2.rectangle(frame,(x,y),(x+w,y+h),(0,0,255),2)

     cv2.imshow("video",frame)

    if cv2.waitKey(1)== ord('q'):

        break

camera.release()

cv2.destroyAllWindows()
```

PROJECT:

```
import cv2



camera = cv2.VideoCapture (0)

car_cascade =
cv2.CascadeClassifier('C:/dspfinal/data/haarcascades/haarcascade_frontalcatface_extended.xml')

while True:


    (ret,frame) = camera.read()

    if ret is True:

        grayvideo = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

        cars = car_cascade.detectMultiScale(grayvideo, 1.1, 1)

    else:

        continue



    for (x,y,w,h) in cars:

     cv2.rectangle(frame,(x,y),(x+w,y+h),(0,0,255),2)

     cv2.imshow("video",frame)

    if cv2.waitKey(1)== ord('q'):

        break

camera.release()

cv2.destroyAllWindows()
```