***Appendix - Source Code for the ArUco-Nav project***

**Project structure**

ArUco-Nav

|\_\_ camera\_calibration\_config

| |\_\_ cameraMatrix.txt

| |\_\_ distortionCoefficients.txt

|\_\_ src

| |\_\_ aruco\_nav\_sim.py

| |\_\_ marker\_orientation.py

|\_\_ .gitignore

|\_\_ README.md

**--camera\_calibration\_config/**

1. **cameraMatrix.txt**

5.096235628673392171e+02,0.000000000000000000e+00,3.247523428613353076e+02

0.000000000000000000e+00,5.087299926026687444e+02,2.479430911954232499e+02

0.000000000000000000e+00,0.000000000000000000e+00,1.000000000000000000e+00

1. **distortionCoefficients.txt**

1.610260403622619241e-01,-2.528776059855320502e-01,1.889781945883286509e-03,-7.598557311923051982e-04,-2.498358492862756408e-01

**-- src/**

1. **aruco\_nav\_sim.py**

# importing required modules  
import math  
import json  
import socket  
import pygame  
from marker\_orientation import getMarkerOrientation  
   
# function to display text on the pygame window  
def blitText(str, color, x, y):  
    text = myFont.render(str, True, color)  
    screen.blit(text, (x, y))  
   
# storing server url  
aruco\_vision\_server\_IP = "192.168.2.2"  
port = 5000  
aruco\_vision\_server = (aruco\_vision\_server\_IP, port)

 # creating a socket instance  
aruco\_vision\_client = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)  
   
# pygame window name  
window\_name = "ArUco-Scanner-Nav-Sim"  
# pygame window size  
width = 1280  
height = 720  
size = [width, height]  
# pygame colors  
BLACK = (0, 0, 0)  
WHITE = (255, 255, 255)  
RED = (255, 0, 0)  
CYAN = (0, 255, 255)  
GREEN = (0, 255, 0)  
   
# initializing pygame  
pygame.init()  
pygame.font.init()  
# creating time and font objects  
clock = pygame.time.Clock()  
myFont = pygame.font.SysFont("Arial", 15)  
# setting up pygame window  
screen = pygame.display.set\_mode(size)  
pygame.display.set\_caption(window\_name)  
   
# establishing connection to ArUco-Scanner vision server  
try:  
   print("Connecting to ArUco-Scanner vision server " + str(aruco\_vision\_server))  
   aruco\_vision\_client.connect(aruco\_vision\_server)  
   aruco\_vision\_client.setblocking(0)  
  aruco\_vision\_client.setsockopt(socket.IPPROTO\_TCP, socket.TCP\_NODELAY, 1)  
   aruco\_vision\_client.settimeout(1)  
except:  
   print("Unable to connect to ArUco-Scanner vision server" + str(aruco\_vision\_server))  
    exit()

   
done = False  
while not done:  
    # setting background color to black  
    screen.fill(BLACK)

   # getting scanned markers  
    try:  
        s = ""  
        # sending request to ArUco-Scanner vision server  
        aruco\_vision\_client.send("g".encode())  
        clock.tick(5)  
        # storing JSON response containing marker data  
        s = aruco\_vision\_client.recv(100000)  
        # populating marker dictionary with received marker data  
        markersDict = json.loads(s.decode())  
    except:  
        continue  
   
    # storing JSON data for all received markers  
    aruco\_markers = markersDict["aruco"]  
   
    # showing connection status and marker count info  
    connection = "> server running at " + aruco\_vision\_server\_IP + ":" + str(port)  
    count = "marker-count: " + str(len(aruco\_markers))  
    status = "> status: OK"  
    blitText(count, WHITE, 5, 5)  
    blitText(status, GREEN, 5, (height - 40))  
    blitText(connection, GREEN, 5, (height - 20))

    # getting individual marker specs  
    for m in aruco\_markers:  
        # getting marker ID  
        marker\_id = int(m["ID"])  
        # getting marker size  
        marker\_size = int(m["size"])  
        # getting marker heading direction in radians  
        marker\_heading = m["heading"]  
        # getting corner coordinates  
        corners = m["markerCorners"]  
        # getting center coordinates  
        Xc = int(m["center"]["x"])  
        Yc = int(m["center"]["y"])

        # computing center coordinates of the top side of the marker  
        Xt = int(Xc + ((marker\_size / 2) \* math.sin(marker\_heading)))  
        Yt = int(Yc + ((marker\_size / 2) \* math.cos(marker\_heading)))

# displaying marker attitude parameters  
         try:  
            xm, ym, zm, roll\_marker, pitch\_marker, yaw\_marker =

getMarkerOrientation(corners, 10)  
         except:  
            pass  
   
        # drawing markers on the screen  
        # drawing bounding box  
        for i in range(4):  
            j = ((i + 1) % 4)  
            # start position coordinates  
            Pi = (int(corners[i]["x"]), int(corners[i]["y"]))  
            # end position coordinates  
            Pj = (int(corners[j]["x"]), int(corners[j]["y"]))  
            # line connecting start and end positions  
            if marker\_id in [25, 400, 750, 900]:  
             # barrier markers  
                pygame.draw.line(screen, RED, Pi, Pj, 3)  
            else:  
                pygame.draw.line(screen, GREEN, Pi, Pj, 3)

# drawing heading line  
        if marker\_id in [25, 400, 750, 900]:  
         # barrier markers  
            pygame.draw.line(screen, RED, (Xc, Yc), (Xt, Yt), 3)  
        else:  
            pygame.draw.line(screen, CYAN, (Xc, Yc), (Xt, Yt), 3)  
        # showing marker ID at the center  
        blitText(str(marker\_id), WHITE, Xc, Yc)  
   
    # updating the full display  
    pygame.display.flip()  
   
    # quitting event loop  
    for event in pygame.event.get():  
        if event.type == pygame.QUIT:  
            done = True  
   
# exiting pygame  
pygame.quit()