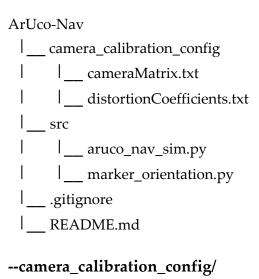
Appendix - Source Code for the ArUco-Nav project

Project structure



1. cameraMatrix.txt

2. distortionCoefficients.txt

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

-- src/

3. 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
 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)
  print("Unable to connect to ArUco-Scanner vision server" + str(aruco_vision_se
rver))
  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()
```