Point of Contact: Michael Sikora ⟨m.sikora@uky.edu⟩

Date: 2018.01.15

Project: Dynamic Microphone Platform Array

Overview

This project contains MATLAB files to define a circular microphone array and perform rotations for simulations. An Object Oriented Programming (OOP) approach was used to define the class Platform in **./OOPstyle/Platform.m**.

Class structure

Platform
loc_center : 1x3 vector of doubles , coordinates of center point **N** : integer , number of microphones **a** : double , radius **quaternion** : 1x4 vector of doubles , rotation quaternion **init_mics** : Nx3 vector of doubles , coordinates of microphones
Platform (*loc_center* : 1x3 vector of doubles, *N* : integer, *a* : double) : Platform **initMics**() **getCenter**() : 1x3 vector of doubles **getMics**() : Nx3 vector of doubles **rotate**(*quaternion* : 1x4 vector of doubles) **orient**(*quaternion* : 1x4 vector of doubles) **eulRotate**(*psi* : double, heading angle in radians; *theta* : double, pitch angle in radians) **eulOrient**(*psi* : double, heading angle in radians; *theta* : double, pitch angle in radians) **centerAt**(*newCenter* : 1x3 vector of doubles) **setRadius**(*newRadius* : double)

Directory Tree

```
./
├── README.md
├── platform3Drotate.m
├── platform3Drotate2.m
├── OOPstyle/
│   ├── Platform.m
│   ├── WallQuaternions.m
│   ├── OOP_rotate.m
│   └── OOP_test.m
└── quaternions/
    ├── quatMult.m
    └── quatRotateDup.m
```

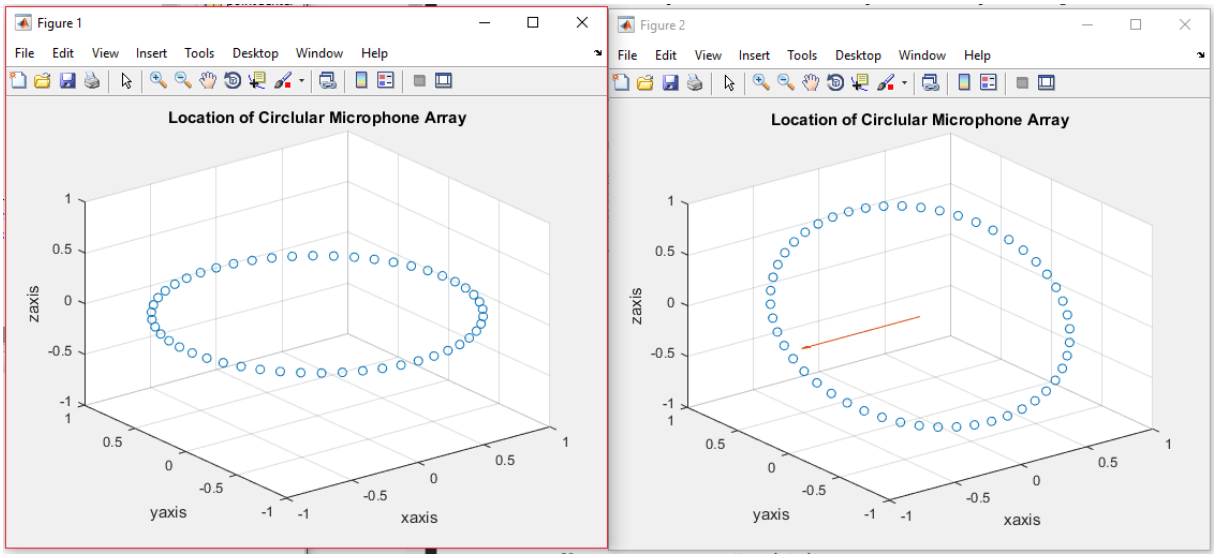
How To Use

A platform is first defined by instantiating the Platform class using a constructor. The x,y,z coordinate vector of the center point, number of microphones, and radius of circular array are the required attributes of the constructor. The method *getMics()* can be called to get the x,y,z coordinates of the microphones. The Platform object is currently defined to be initially oriented as it would be on the floor of a room. The constructor calls the method *initMics()* to define the x,y,z coordinates in this orientation, as well as the initial quaternion of this orientation. The method *rotate(q)* multiplies the platforms quaternion attribute by **q**. Here is an example of the syntax:

```
pointCenter=[0 0 0]; N=50; radius=1; % Define attributes
p1 = Platform(pointCenter,N,radius); % Construct object
% Get initial microphone coordinates
loc_mics_initial = p1.getMics();
quaternion = [0.9239 0.3827 0 0] % define a quaternion
p1.rotate(q); % rotate using quaternion
% Get rotated microphone coordinates
loc_mics_rotated = platform1.getMics();
```

Example

The below figures show the result of **./OOPstyle/OOP_test.m**. The circular array is defined with 50 microphones, a radius of 1 distance unit, and a center point at the origin. The quaternion used to rotate the array is defined by a -$\pi/4$ angle rotation about the -y axis.



Point cloud systems were also initially tested. Run **platform3Drotate2.m** to view a bounded plane rotated in 3D space. Run **platform3Drotate.m** to view another planar point system bounded by a circle rotated in 3D space. This point cloud method was used in **./OOPstyle/WallQuaternions.m** to plot a set of points on a virtual wall.