EE587 Introduction to Robotics

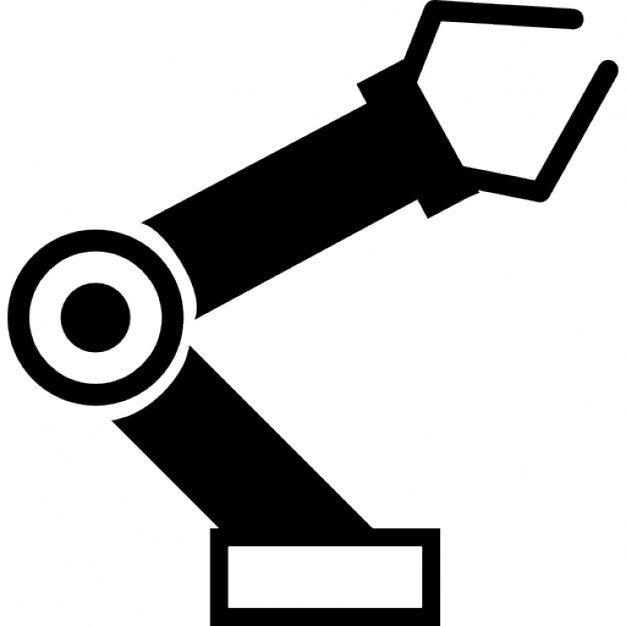
Submitted to:

Prof. Dr. Aydan Erkmen  
Student:

Seyit Yiğit SIZLAYAN

1876861

2nd Term Project Report



Contents

[Project Definition 2](#_Toc472919481)

[Assumptions 2](#_Toc472919482)

[DH Parameters 3](#_Toc472919483)

[Kinematics 4](#_Toc472919484)

[Dynamics and the Jacobian 6](#_Toc472919485)

[Control 9](#_Toc472919486)

[Conclusion 17](#_Toc472919487)

[References 18](#_Toc472919488)

# Project Definition

Two drones play the bandeneon. The sound is produced by the amount of air pushed in certain trajectory.

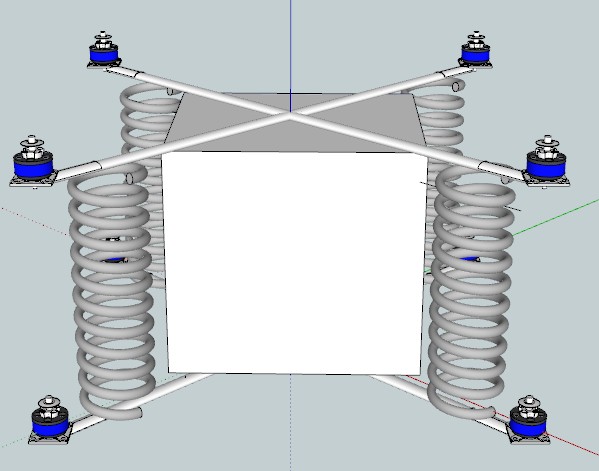


Figure 1 Basic 3D model of the system, 2 quadrotors with spring-bandeneon in between

# Assumptions

1. The drones and their motor-propeller systems are ideal and works linearly i.e the motors can give any thrust force up to max speed, no cogging torque on the motors.
2. The drones are classical, 90 degree-quadrotors whose consecutive motors are rotating in the opposite direction which results in no net torque at the center.
3. The quadrotor can be modeled as two thin, intersecting, uniform rods with point like motors at the end of the rods, causing thrust force. The bandeneon can be modeled as identical springs which are connected at the end of the rods as can be seen in figure 1. The sound is obtained by changing the volume of the rectangular prism whose edges are also the lengths of the springs.For the clearance of the visual, they are drawn seperately but they actually coincide.
4. The springs squeezes only in the direction of its vertical axis, they cause no torsion and they do not bend.
5. In order to play a bandeneon there is no need for the yaw motion of the quadrotors because it only generates torsion, it does not change the volume of the bandeneon. So, yaw motion generate no sound in the system. Also, due to the 2nd assumption, there is no torque which can cause yaw motion, so the efect of the yaw motion is negligible. In the control of the system, controller will oppose the yaw motion.
6. The edges of the bandeneon can stretch infitely and shrink down to zero.
7. Even though the air friction is a required for the thrust, the air friction due to bandeon and other components are negligible. Also the air around the quadrotor is uniform and stationary, i.e no disturbing forces.

# DH Parameters

In order to be able to generate DH parameters, we need to identify the end effector. In this case, seperating the system into two will be more appropriate because each quadrotor is controlled seperately and the distance between the connected links is what we need to control since the output will be the difference in the volume of the bandeneon. In this system, there are 8 end-effectors which are the ends of the rods, since the distances between links are changed by them.

Since quadrotors are identical and their centers overlaps in x and y directions intially, the DH parameters generated for one can be used in the other by changing the z-axis component of the translational part namely t34.  
 Since there are multiple end effectors, DH parameters will be generated for the frame which is just before the end effectors, namely the body frame for quadrotor design.

A drone is a device which has 6-DOF. 3 of them are prismatic joints and 3 of them are revolute joints. Using Euler angles,

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
| 1 | 90 | 0 | d1🡪x | 0 |
| 2 | 90 | 0 | d2🡪y | 0 |
| 3 | 90 | 0 | d3🡪z (For upper drone there will be +Lb) | 0 |
| 4 | 90 | 0 | 0 | 🡪 |
| 5 | 90 | 0 | 0 | 🡪 |
| 6 | 90 | 0 | 0 | 🡪 |

\*Lb is the one edge length of the initally-cubic-bandeneon.   
🡪This can be thought as a spherical joint moving with PPP manipulator.

# Kinematics

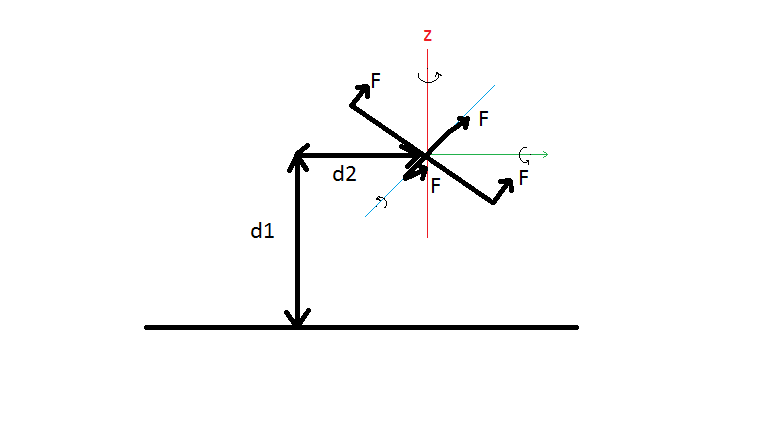


Figure 2: Planar viewof the quadrotor system (For the sake of figure, it was drawen tilted. Frame is for horizontal drone configuration)

Z’Y’X’ Euler Rotation matrix is:

And due to prismatic joints at the origin of the fixed frame.

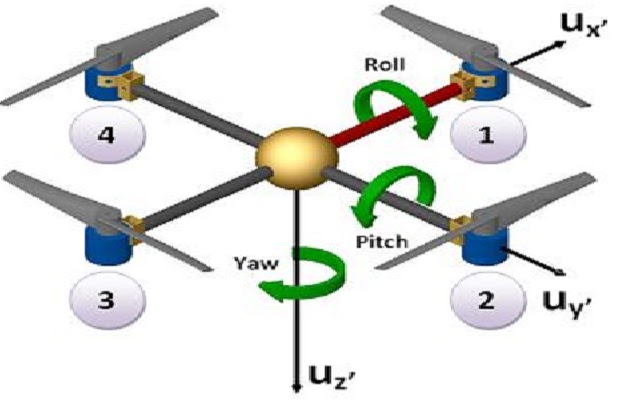


Figure 3: Euler angles with respect to body frame, obtained thrust forces from motors M1,M2,M3 and M4 [1]

The homogeneous transformation matrix of body frame(origin is at the center) to fixed frame (origin is at the point where flight begins) will be:

However, this does not give us the pozition and orientation of the end effectors, this transformation gives us the body frame which has a origin at the center of the quadrotor with respect to the fixed frame. For the end effectors (there are 4 of them for each drone), we need to translate the frame in the ± x and ± y with respect to the body frame.  
So by post-multiplying the T with Trans ±Xb and Trans ±Yb , we can obtain the resulting end-effector frames.

For the 1st motor frame which is one of the end effector condiguration of the lower drone:

For the 1st motor frame of the upper drone which is initially +Lb higher than the lower drone with respect to the z-axis of the fixed frame.

# Dynamics and the Jacobian

As stated above, since a drone is 6-DOF device its joint matrix will be :

ψ is the yaw angle around the z-axis, θ is the pitch angle around the y-axis and φ is the roll angle around the x-axis.

Since controlling the drones is easier than directly trying to control the end effectors, I will generate the dynamics for the quadrotors which is connected to other drone with springs. By doing so, only the potential energy term will differ from the dynamics of a single quadrotor.

The Lagrangian in this system consists of 3 parameters:

where m is the mass of the drone and I is the inertia matrix.

For the rotational kinetic energy:

This equation is valid since rotation on a drone is independent from the X.

Which results in

And since the drone is modeled as thin rods:

Substituting the relation above in the rotational kinetic energy formulation:

Lagrange equation with respect to q:

Where FX is the translational forces on the quadrotor caused by the thrust force of the propellers and τ is the moments in the direction of yaw, pitch and roll with respect to the center of the quadrotor.

Since initially drone is horizontal and all the forces acting in the +z direction. Since there is 2 drones, net force generated by the springs become zero:

and

Considering the torques:

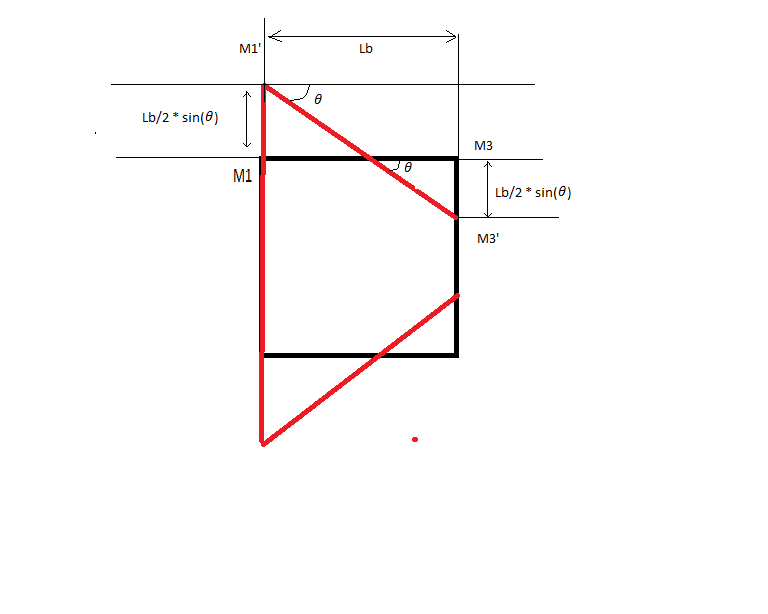


Figure 4: The changes in the spring lengths. Initially the system is in black configration and motors are mounted at M1 and M2 points. When the pitch motion occurs, the configuration becomes the red one.

And also this causes change in the potential energy:

The lagrangian can be seperated in to two in terms of X and q since they contain no cross terms in the kinematic energy combining X and q [2]. [1]

Then

In terms of q:

In short, the overall dynamic model for the system:

And force of gravity with spring bandeneon spring force:

The overall system mechanic is:

These quations is applicable for both upper and lower quadrotors. Since we are calculating the spring force due to bandeneon using the world frame distances, it can be put into account directly.

# Control

For the control of the system, I first considered implementing the dynamics in MATLAB then simulating inverse dynamics control on it. But then I decided to use professional robotic simulator for this purpose because my dynamic model has lots of flaws: for example I do no know the inertias Ixx , Iyy , Izz or the mass of a quadrotor mquad. And in addition to them, inverse dynamics control would not be robust enough to compansate the actual air thrust applied to the lower quadrotor in real world. Even though it works in matlab, it probably would not work in actual world. From this point on, the assumptions I made above will not be true because I implemented the control section directly in the simulation.

Because of the reasons I mentioned above, I searched for a robotic simulation which has decent quadrotor design and require less prior experience. I found V-REP(Virtual Robot Experimentation Platform) Robotic Simulation of Coppelia Robotics [3] whose educational version is free for the students. In addition to that, it can work on Windows, macOS and Ubuntu, it has both 32 and 64 bit versions. The most important part for me was the simulation environment can be controlled by python, Java, C, C#, C++, Matlab and Lua. It has remote api libraries for all the languages mentioned. There are 5 different types of physics engines used on it and it allows the user to choose one of them for the simulation. In the project I preffered using Lua scripting language which is the native script language supported by V-rep (no remote api required) and Bullet 2.78 pyhsics engine.

V-rep has a quadrotor model based on commercial product Parrot AR-Drone [4].

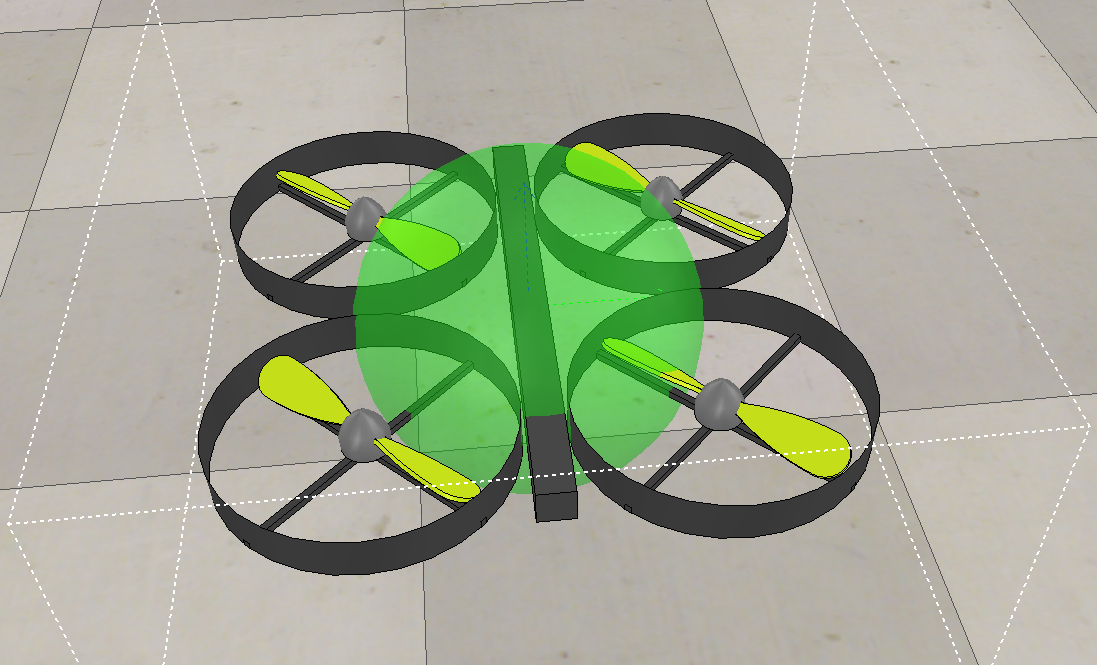


Figure 5: V-rep model of the AR drone on the left and commercial AR-drone quadrotor

The most problematic part for the simulation was implementing the bandeneon as springs. There are spring damper models on the simulation but they were not suitable for the system because of their lenghts. So, I changed the dynamic model script of the propellers to generate a force as a function the distance between the quadrotors and add this force as thrust for the lower quadrotor or subtruct it for the lower quadrotor. I left the spring constant as parameter so I can also observe the effect of the spring constant. The effect of it can be seen from the figure below, the initial conditions and desired locations are the green spheres. As can be seen form the figure on the left, high spring coefficient (k=1N/m) pulled the lower quadrotor too much and it cannot move to its desired location anymore. The figure on the right belongs to the lower spring coefficient(k=0.01N/m).

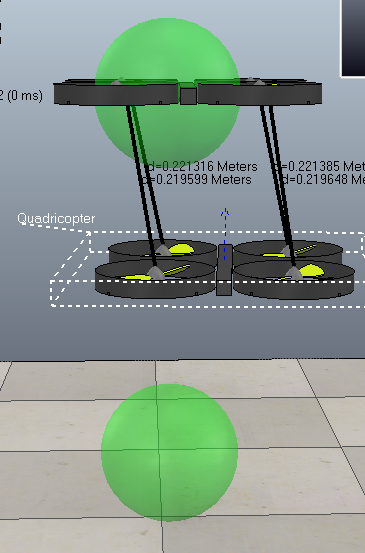
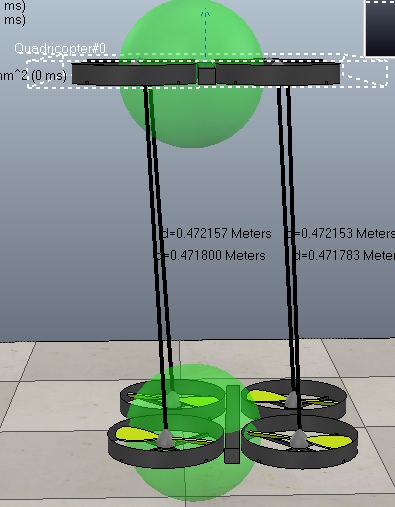
 

Figure 6: High spring coefficient effect on the left, low spring coefficient effect on the right

As also can be seen from the figure above, the location of the upper quadrotor is not affected by the spring coefficient. That is because, the propellers of the quadrotor rotates only in one direction, meaning that they can generate a force only in the +Z direction. When the spring coefficient is increased, the upper quadrotor is pulled down but it can compansate the pulling force by increasing the propeller speeds. But the lower quadrotor cannot generate a force against the force pulling it up. This can be compansated by increasing the mass of the quadrotor or the making the propellers reversible but I could not change the model accordingly. When I increased the mass, it changed all the dymamic model. If the propellers become reversible, since they have friction on them, they could not work properly at lower speeds.

The code generating dynamic behvious of the spring can be seen below:

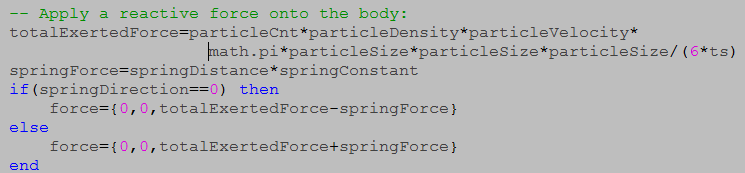


Figure 7: Spring force dynamics applied to the propellers,spring direction is 0 for the upper quadrotor and 1 for the lower quadrotor.

After implementing the springs on the quadrotors, I began to implement controllers. The model itself had primitive P controller for altitude(Z), X, Y and yaw. I developed my controller based on them.

I thought implementing force feedback(feedback from spring) and inverse dynamic controller based on my model but I could not do that because actual quadrotor model were different than mine and system become unstable with that controller due to uncertainity of the propellers, they have gyroscopic effect, generate air flow and their moment of inertia was not included in my model.

So I generated dicrete PID controller which is called everytime the simulation iterates. The code segments can be seen from the figure below. The particleTargetVelocities are the required propeller velocities to generate required torques and thrust forces. The other variables are the p,i,d parameters for the attitude control. ‘p’ prefix (as in pAlphaE) represents the previous iteration values. As a convention of the V-rep simulation, I used alpha for Euler angle around X-axis and Beta for the Euler angle around the Y-Axis.

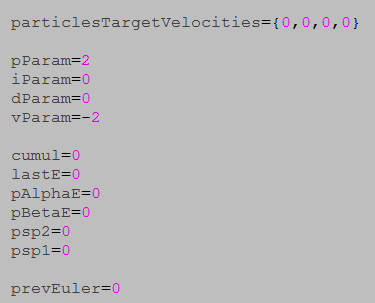


Figure 8: Control constants and variables

The main purpose of the altitude controller is making the error between the target object position’s altitude and altitude of itself zero. For this, error e is calculated for vertical control and fed back as thrust variable to the propellers. The parameters for this are p=2, i=0, d=0. The other constants like 5.335 or vParam were the constants given by the simulation itself.

For horizontal control, first I needed to get the rotation matrix because the direct position control cannot be done in X and Y directions. For this I first calculated the required roll(alpha) and pitch(beta) angles then using PID control I controled the velocities in the X and Y direction with respect to the quadrotor frame. I controlled the speeds because the quadrotor kinematics allows motions only in the roll and pitch direction, the velocities in X and Y are linear functions of pitch and roll respectively. The PID parameters was p=0.25 and d=2.1.



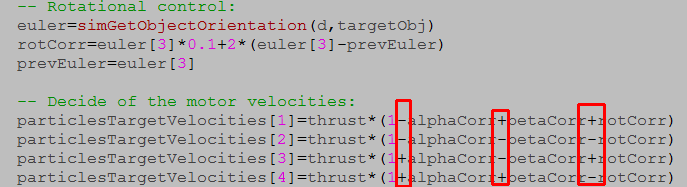
Lastly, there were a yaw controller implemented in the model, I did not change it because I do not want a motion in yaw direction as my control policy, the controller prevents this motion. After calculating correction variables, the required propeller speeds are calculated. As can be seen from the code below, each propeller is affected from thrust differently because their effect on motion is unique.  


Figure 9: Yaw controller and the propeller speed calculation. The effect of the propeller positions are shown in red rectangles

After tuning the PID controller, the model follows a “dummy” (V-Rep object which represents a coordinate frame with its origin and rotations). The output of the altitude(Z), X and Y control responses can be seen from the graph below:



Figure 10: Altitude response of the upper quadrotor



Figure 11: X control response of the upper quadrotor



Figure 12: Y control reponse of the upper quadrotor

Steady-state error due to spring can be seen at the end of the graph. I tried to minimize it, but since the output of the system will be the difference of the positions of the 2 quadrotors, the steady state term will cancel at the output since both quadrotors behaves identical in X-Y control.

Since I could control a quadrotor in 3D(X,Y,Z), I implemented a path following algoritm to be able to play a bandeneon. There was a path facility implemented in V-rep which generates intepolated points between the coordinates taken from the user.

At this point, I have to explain how a bandeneon is played by amateurs. One of the hands is used to supply the required air while the other is used to generate the noise. So to mimic this behaviour, I generated a circular path for both upper and lower quadrotors but the radius of the lower quadrotor is much smaller than the radius of the upper quadrotor. By doing so, lower quadrotor behaves as a noise-generating-hand while the upper supplies air. I preffered this configuration because upper quadrotor is more flexible compared to the lower quadrotor due to the spring effect of the bandeneon.

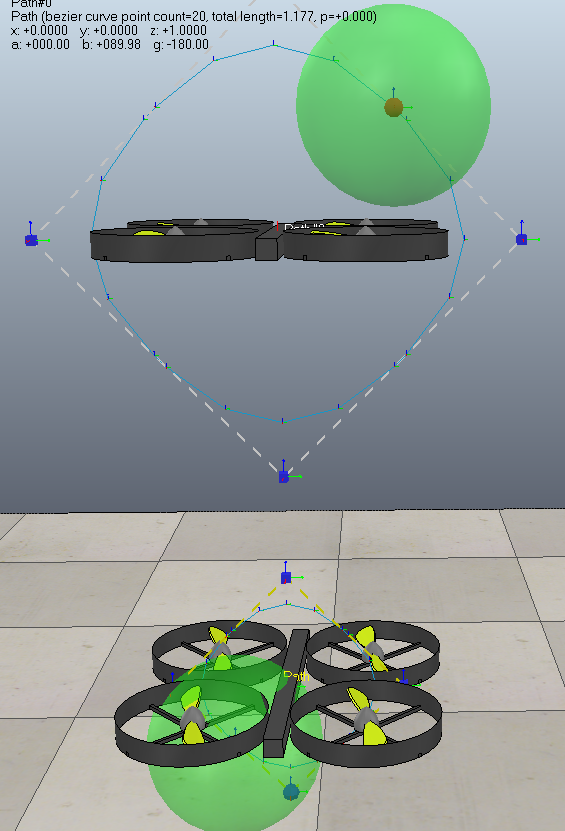
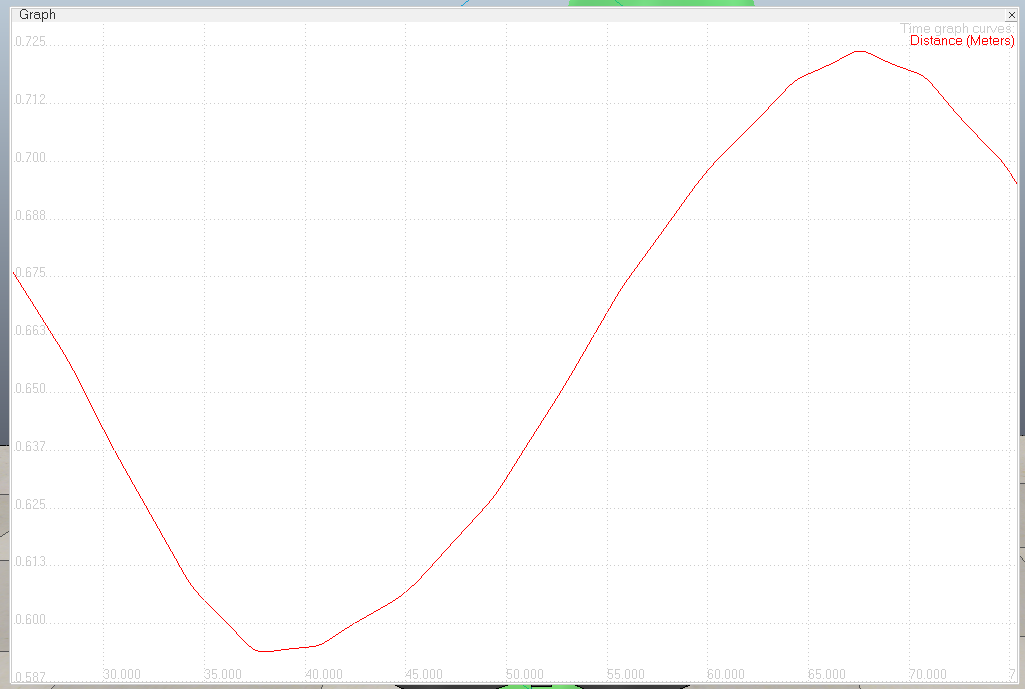
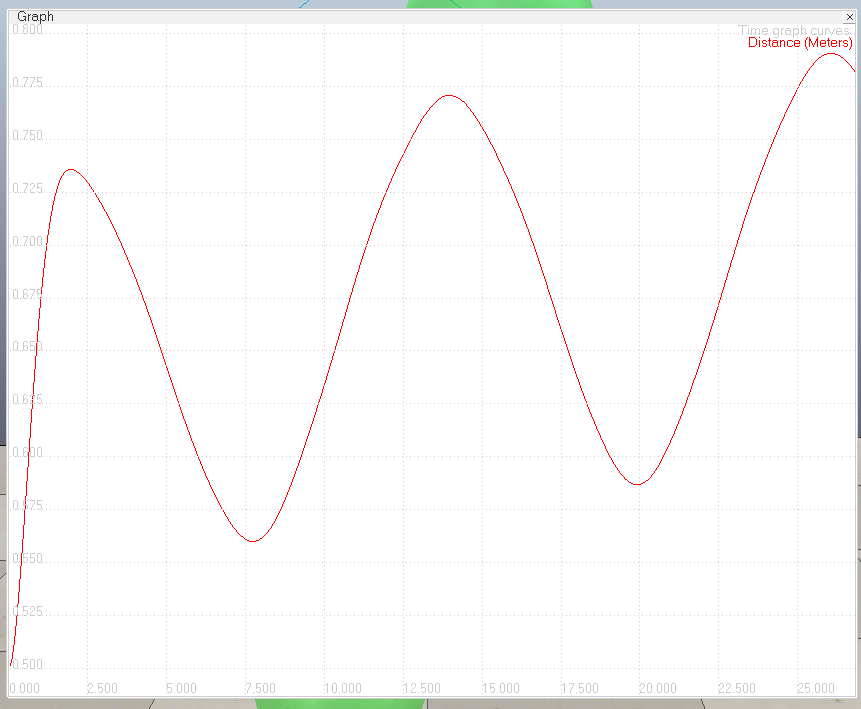
The path scheme can be seen below as thin blue circles on the quadrotor models.  


Figure 13: Quadrotor paths, upper has bigger radius to supply air while lower has small radius to generate noise frequencies much easily

The green spheres are the target positions of the quadrotors which are moving on the path “dummy”. Since the desired position is changing, the qudrotor controllers are following them, by doing so they also follows the path.  
 Since our main purpose is generating a noise as a function volume of the bandeneon, the system must move with a changing frequency. I implemented this behaviour by changing the rotation speed i.e the period of the target position on the path. Again, the upper quadrotor have small speed/ high period, while the lower quadrotor moves with high speed/small period.

I generated the output graph by measuring the distance between upper and lower quadrotor in the Z axis with respect to the world frame since the volume is actually the function of this distance because the base area is constant in this system.  
 Some of the output results can be seen below:





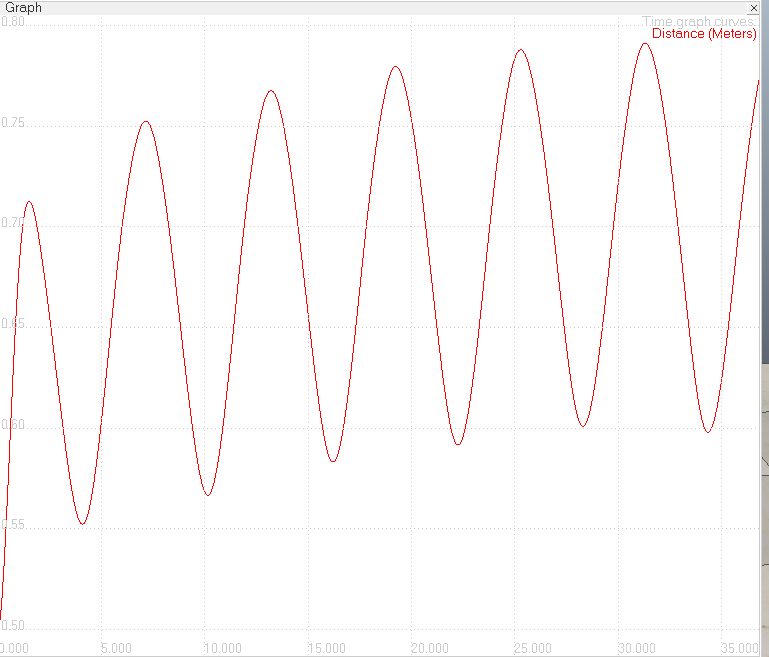


Figure 14: The output of the system with 3 different path following period(0.1,0.05,0.01 s), from highest to smallest

By changing the radius of the path and path-following period one can play the bandeneon with this system.

# Conclusion

In this project, I generated the dynamic model of the bandeneon-playing, 2-drone system and controlled it with PID controller to follow a closed-path with changeable period.  
 This project allowed me to learn generating D-H parameters easily and understand the dynamic behaviour of a quadrotor system. It also gave me chance to use LUA scripting language and tought me how to use a professional robot simulator V-Rep.

I had implemented discrete PID controllers before, mostly to control the position, but in this project I used PID controller for the velocity control. I also gain the experience on path planning and tracking.  
 All in all, this project was very instructive about robotic systems and helped me to clearly understand the topics covered in the class.

# References

|  |  |
| --- | --- |
| [1] | K. Ahmedi, «http://www.microsolution.com.pk/,» 2013. [Çevrimiçi]. Available: http://www.microsolution.com.pk/quadcopter-in-pakistan/. [Erişildi: 12 2016]. |
| [2] | A. E. D. L. R. L. C. P. Luis Rodolfo García Carrillo, Quad Rotorcraft, New York: Springer, 2013. |
| [3] | C. Robotics, «VREP,» Coppelia Robotics Inc., 23 01 2017. [Çevrimiçi]. Available: http://www.coppeliarobotics.com/. [Erişildi: 23 01 2017]. |
| [4] | P. Inc., «AR drone,» Parrot Inc., 21 01 2017. [Çevrimiçi]. Available: https://www.parrot.com/fr/drones/parrot-ardrone-20-elite-edition#parrot-ardrone-20-elite-edition. [Erişildi: 21 01 2017]. |