MASSACHUSETTS INSTITUTE OF TECHNOLOGY

DEPARTMENT OF AERONAUTICS AND ASTRONAUTICS: SPACE SYSTEMS LAB

Prototyping Robotic Manipulators For SPHERES

Lisandro Jimenez, Edward Lopez, Duncan Miller

August 12, 2014

1 INTRODUCTION AND BACKGROUND

SPHERES is a project that is based around small satellites(8 inches in diameter) that use CO_2 thrusters and sensors to maneuver with other satellites. SPHERES are used to explore possibilities in spacecraft-to-spacecraft maneuvering and communication. SPHERES is an acronym for Synchronized Position Hold, Engage, Reorient Experimental Satellites. These satellites, aboard the International Space Station, provide a low risk experimental platform to test out different algorithms and techniques that would otherwise be difficult/expensive to test. On the ISS, these experiments can be carried out in microgravity in full six degrees of freedom. These experiments are used to explore experimental space technologies and algorithms, such as spacecraft flight formation and, in this case, robotic arms.

Having a robotic arm attached to satellites/spacecraft can be very useful, but can introduce new problems. The application of these arms can range from helping astronauts on spacewalks to autonomously constructing large structures in space. One problem is that

when the arm is moved, the entire satellite moves in response.

In this project, we used LEGO MindStorms, as it provides an easy-to-work-with solution to our need to iterate rapidly. Currently, there is not a testbed for testing algorithms for spacecraft with robotic arms, and this project aimed to create one. For this project, a team of three UROPs[1] worked to develop and refine the robotic arm, develop the hardware and software involved in communicating between the arm and SPHERES, and to develop simulations of the SPHERES-arm interaction.

2 Literary Research

At the start of this project, we looked at what had been done before so that we could build off of that and create a project that would contribute to the field of space robotics. The information we compounded is located in the appendix of this report, which contains a matrix of what has and has not been done as well as a list of academic papers pertaining to the subject matter. The essence of what we learned from this was that there is quite a bit of theoretical research done but little experimentation has occurred. With that, we figured it would be best if we made an arm that it could move in 3D space and be able to simulate the system (SPHERES, Halo and the Arm) dynamics. We also thought the final, end-all be-all goal would include using Vertigo[2] to track a moving object, then using the arm to grab it, and finally using thrusters to bring to system to a stop. We never got to this though.

3 Simulation and Planning

Our first step was to simulate the dynamics that would be experienced when the arm was manipulated, and to be able to know what positions to put each joint of the arm in order to be able to get the end-effector in a desired position.

[1] Undergraduate Research Opportunity Program
[2] Other SSL project involving cameras for vision tracking. See Vertigo Homepage for more info

3.1 Inverse Kinematics

Inverse Kinematics is used to determine how a structure moves and rotates to reach a desired point. For the arm specifically, we needed to know how much to rotate each motor to reach a desired location. In 2D, the most efficient way of doing this was with circles. With a two-joint arm, all that is needed to do is draw one circle with a radius of the length of the first arm segment centered at where the arm begins and draw another circle with a radius of the length of the second arm segment centered at the desired location. Where these circles intersect is the location that the second joint (beginning of the second arm segment) needs to be in order to reach the desired location.

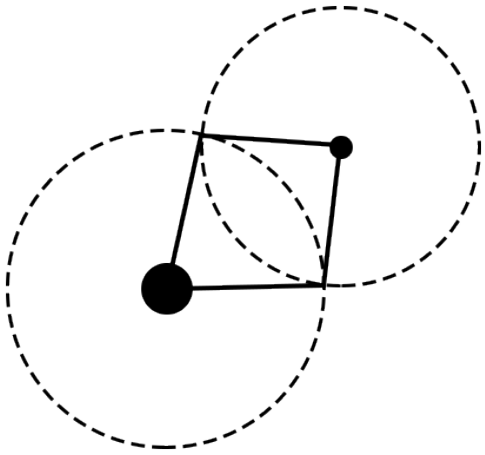

Figure 3.1: Circle Method for inverse kinematics

The same method can be used for greater degrees of freedom(more arm segments). There will be infinite solutions, however, as opposed to two, due to the nature of two dimensional space. Figure 3.2, showing one possible solution, illustrates this concept. A limited amount of circles are shown in the figure for simplicity, but in actuality there are an infinite amount of circles around the lower large circle.

The Circle Method is easily extended into the third dimension.

To implement the Circle Method in software, we used Matlab. To do this, we used *fsolve*, a system of equations solver. The two equations were the two equations of the circles, which held their radius and center point. The variables we solved for were x_1 and y_1, the location of the second joint. From there, we did simple trigonometry to solve for the angle the first arm

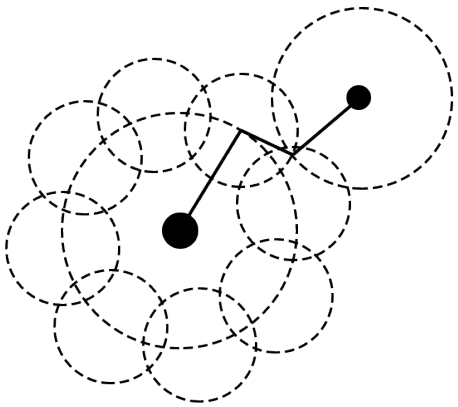

Figure 3.2: Circle Method for 3 DOF

made with respect to the x-axis and the angle the second arm made with respect to the first segment. This can be expanded in two ways: 3D and more arm segments.

For more arm segments, we just need to have more circles, one for every segment. The thinking here is that you the first segment draws out one circle. The start of the second segment has to be located some point on that circle, so it is assigned an arbitrary point (x_2, y_2) on the first circle. Centered on that point, it draws out a circle with a radius of its length. The equation of that circle will look like $(x_2 - x_1)^2 + (y_2 - y_1)^2 = L_2^2$, with (x_2, y_2) representing a point on the rim of the circle. The third segment has to begin somewhere on the second circle, so it is given an arbitrary point (x_2, y_2) and draws a circle centered at that position. It's equation looks like $(x_3 - x_2)^2 + (y_3 - y_2)^2 = L_3^2$. This cycle continues until the nth segment centered at (x_{n-1}, y_{n-1}) draws a circle of its own. We want the desired location to be on that circle, so it's equation is $(x_n, x_{n-1})^2 + (y_n, y_{n-1})^2 = L_n^2$, with (x_n, y_n) representing the desired location. From there, you let *fsolve* do the rest of the work.

$$\begin{aligned}(x+y)^3 &= (x+y)^2(x+y) \\ &= (x^2+2xy+y^2)(x+y) \\ &= (x^3+2x^2y+xy^2)+(x^2y+2xy^2+y^3) \\ &= x^3+3x^2y+3xy^2+y^3\end{aligned} \tag{3.1}$$

With 3D, it gets a bit more complicated. Currently the motors we are using only have

one degree of rotational freedom, which greatly limits them. If they had three degrees of rotational freedom, it would be simple: we would use spheres instead of circles in the method described above. But alas, life is not that easy. This is because in our design all of the arm segments lie on the same plane, which would not be considered when using spheres. That being so, we have not come up with an exact solution but we an idea on how to approach this. We were thinking of using a change

3.2 Dynamics Simulation and Visualization

In order to predict the dynamics of the system, we came up with a formula resulting from Newton's 2nd Law. This formula describes how a two-segment system will move given the moments of inertia of both segments about the pivot point and at least one displacement angle is known.

$$\frac{\theta_1}{I_2} = \frac{\theta_2}{I_1} \tag{3.2}$$

Where I_1 and I_2 are the moments of inertia of each of the respective segments, and θ_1 and θ_2 are the respective displacement angles.

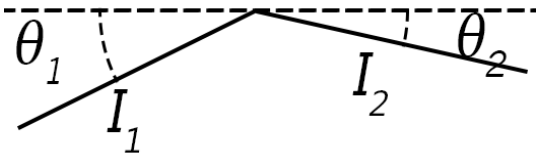

Figure 3.3: Angle Displacement Method

We created a graphical simulation in C++ using the SFML graphics library in order to visualize the method we came up with to predict arm dynamics. We created a class to represent an arm-SPHERES system. This class in turn contained three "arm segment" class objects(one for the SPHERE, two for each joint of the arm). It then became easy to manipulate the joints of the arm and find out information of the system. Because of this, we were able to implement a brute force search method to find the motor positions that would result in the arm reaching a certain spot. This is independent of graphics, which allows us to run it on, for example, Halo. Using SFML, we took this information and drew basic shapes to represent each of the arm segments and the spheres.

3.3 Heading on level 2 (subsection)

Lorem ipsum dolor sit amet, consectetuer adipiscing elit.

$$A = \begin{bmatrix} A_{11} & A_{21} \\ A_{21} & A_{22} \end{bmatrix} \tag{3.3}$$

Aenean commodo ligula eget dolor. Aenean massa. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Donec quam felis, ultricies nec, pellentesque eu, pretium quis, sem.

3.3.1 Heading on level 3 (subsubsection)

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Heading on level 4 (paragraph) Suspendisse vel felis. Ut lorem lorem, interdum eu, tincidunt sit amet, laoreet vitae, arcu. Aenean faucibus pede eu ante. Praesent enim elit, rutrum at, molestie non, nonummy vel, nisl. Ut lectus eros, malesuada sit amet, fermentum eu, sodales cursus, magna. Donec eu purus. Quisque vehicula, urna sed ultricies auctor, pede lorem egestas dui, et convallis elit erat sed nulla. Donec luctus. Curabitur et nunc. Aliquam dolor odio, commodo pretium, ultricies non, pharetra in, velit. Integer arcu est, nonummy in, fermentum faucibus, egestas vel, odio.

4 Lists

4.1 Example of list (3*itemize)

- First item in a list

  - First item in a list
    - First item in a list
    - Second item in a list
  - Second item in a list
- Second item in a list

4.2 Example of list (enumerate)

1. First item in a list
2. Second item in a list
3. Third item in a list