**Pre-class Video Journals**

Video: Lecture 1 | Intro to Robotics

**Key Terms:**

**Kinematics:** The motions made by coupled objects when moving to a position.

**Dynamics:** The forces applied by the joints of coupled objects when moving to a position.

**Motion Planning:** The path a robot will take to reach a position.

**Motion Control:** The movement of the robot made by the various motors in its joints.

**Force Control:** Controlling the amount of force applied when moving or manipulating objects in space.

**Redundancy**: When a robot has many different ways of reaching a position due to the several degrees of freedom allowed to the robot.

How do we determine the position of a robot? If we use GPS, this would only work for a robot with only a few degrees of freedom because for each degree of freedom, a GPS device would be used. So, there would be 60 GPS devices for a robot with 60 degrees of freedom, and that would get too costly and complicated. One idea to determine location of the robot would be to identify nearby objects using vision systems. After figuring out how to locate the robot, we need to find out where the position and orientation of the robot’s external “limbs” if applicable.

In a humanoid robot, when one rigid body part moves, the other body parts move as well due to the coupling of the joints. Moving any body part generates **kinematics and dynamics** and this information, such as torque, orientation, acceleration, etc., must be known for the robot to function in the way the engineer wants it to correctly. **Forward kinematics** gives you the location of the hand or whatever the end-effector element is, and **Inverse kinematics** gives you the joint angles of the coupled rigid bodies given the position of the end-effector element. With this knowledge, you can control the joints to move to the appropriate joint positions to fit the position of the end-effector element. Inverse kinematics becomes more and more difficult the more degrees of freedom are available to the robot. Any robot with 6 or more degrees of freedom has an infinite number of ways to reach a position anywhere in its reach.

Robots were originally thought of as complex machines in enclosed environments doing the same, very specific tasks over and over again. But as we continue to further robotics, robots are now thought of as complex machines capable of a variety of tasks in many applications and they don’t have to be in an enclosed environment. People are trying to further robotics to make them closer to an artificial human, capable of decision-making, carrying and manipulating physical loads, and able to work in mostly any terrain just like humans can. Not just humans, but any life in general such as dogs and dragonflies. Many major league companies, such as Sony and Honda, have functioning humanoid robots that can do things such as dancing, flipping, and balancing, a simple task for people, but much more complex and astounding feat to accomplish in a robot.

One advancement in robotics is haptic feedback. It is why a surgeon can operate on a person using a robot. The surgeon would “feel” the resistance of whatever the robot is touching through haptic feedback and can feel the resistance of the scalpel as it cuts through flesh to precisely make an incision.

An interesting idea when making a robot is considering if the robot should be more biologically based or more functionality based. Take for example a dog. If someone wanted to make a robot dog, would they like the dog to look like and bark, whimper, howl, pant, and play like a biological dog, or would they want the dog to excel at activities a dog can do and beyond such as, running, jumping, attacking, protecting, and add more functionality that a normal dog can’t do like carry heavy loads?

A great idea with robotics is to make a robot that can extend the capabilities of a human to realms much further beyond what a normal human or even a human at the pinnacle of human strength can do with the use of an exoskeleton. People have come far in developing techniques to find the best path for a robot to take using **Motion Planning** by observing the natural way people and other forms of life interact with their environment.

These exoskeletons could be used in construction where massive machines are required to lift heavy metal beams or even to lighten the load for back-breaking work faced by construction workers. Another application would be in emergency services where people are stuck under rubble or a massive flood where water currents are too strong to cross. Law enforcement could possibly use these exoskeletons to access areas usually too reinforced to access by normal means or to protect themselves from incoming bullets. There are extreme benefits if humanity could crack the code on how to make a robot that could act and decide like a human can, which is why robotics tends to develop further and further towards mimicking a human.

As with any major development that could greatly impact humanity, safety is a great cause of concern. If you create an extremely powerful exoskeleton that can lift extremely heavy loads without issue, there’s a lot of power in that exoskeleton that can cause major accidents or even be used for crime. One can imagine how much havoc can be caused by someone with evil intentions and supernatural strength, all you have to do is watch a movie. **Motion and Force Control** is real issue here, as in how will the exoskeleton know when to use great force and when not to. This is only the issue with exoskeletons. Imagine what it would be like to have robot that mimics a human so well they’re hardly differentiable. If they’re exactly like us, then they can act exactly like some other not-so-good humans and be used detrimentally as well, intentionally or unintentionally.

**In-Class Assignment Unit 0**

classdef add\_class

%This is a simple class where I show you how to do

%Object Oriented Programmingin Matlab.

%Usage:

%a = add\_class (5,5);

%a

%a = a.X(10); % try it withoug the a = and see what happens

%a

%a.add()

%a.view()

properties

x; % This is where the x value will be stored.

y; % This is where the y value will be stored.

z; % This is where the z value will be stored.

end

methods % All the functions that act on the data.

%Constructor class called when you setup the class

function obj = add\_class(x,y,z)

obj.x = x;

obj.y = y;

obj.z = z; % adding another variable to be utilized in this class

end

%The sum function.

function sum = add (obj)

sum = obj.x + obj.y + obj.z; % The extra z variable perform fine in this function, however it doesn't make much sense in the next functions

end % unless the user is allowed to choose which variables to perform on.

%The subtract function.

function answer = subtract (obj)

answer = obj.x - obj.y - obj.z;

end

%The multiply function.

function answer = multiply (obj)

answer = obj.x \* obj.y \* obj.z;

end

%The divide function.

function answer = divide (obj)

answer = obj.x / obj.y / obj.z;

end

%The view function

function view (obj)

s = sprintf ('x = %.2f y = %.2f z = %.2f', obj.x, obj.y, obj.z);

disp (s)

end

%Function to change up the x value

function obj = X (obj, x)

obj.x = x;

end

%function to change the the y value

function obj = Y (obj, y)

obj.y = y;

end

%function to change the the z value

function obj = Z (obj, z)

obj.z = z;

end

end

end

**In-Class Assignment Unit 1**

1. In a sentence or two, define kinematics, workspace and trajectory.

Kinematics is the motion of an object and its corresponding orientation and position. Workspace is the area the robot is around to work on. Trajectory is the path a robot will take when deciding to move.

1. In a sentence or two, define frame, degree of freedom and position control.

Frame is the structure of the robot which gives it its rigidity and the joints that allow it to manuever. Degree of freedom is a description for a joint having a way of maneuvering. Position control is the effective movement of the robot, particularly its joints, into specific positions by use of PID controllers, etc.

1. Explain in a few short sentences at least 3 different applications covered in the introductory unit.

One application of robotics in the introductory unit was in the medical industry for a doctor to operate on a patient remotely. Another application was in building an exoskeleton for a human to effectively allow a human to do superhuman tasks effortlessly. Another application was in the landscaping industry where a robot could cut down tree branches from the ground up.

**Video Journal 2**

Video: Lecture 2 | Intro to Robotics

**Key Terms:**

Kinematics:the models that we have to describe the robot’s position by finding the robot’s frame, links, and joints.

Revolute Joint: Joint that allows you to rotate about a fixed axis

Prismatic Joint: Joint that allows you to translate about a fixed axis

Generalized Coordinates: A set of independent configuration parameters

Degree of Freedom: Number of generalized coordinates

**Summary:**

The first part of the lecture introduces the idea of special descriptions, that is, how to describe and assign parameters to an object in space. This is accomplished by representing the parts of a robot with respect to its other parts. For example, to find the end-effector of a robotic arm, one can determine its position relative to the links, joint angles, and base of the arm. He goes on to explain the different parts in a robotic arm with the main components being the base, the two types of joints which are either revolute joints or prismatic joints depending on whether they allow a rotation or transformation movement, each link in the arm that is coupled by joints, and the end-effector.

The lecture focuses on the idea of generalized coordinates and degrees of freedom which is determined by taking the number of moving links and subtracting that by the number of joints which provide only one degree of freedom. For the purposes of this lecture, each joint only bring with it one degree of freedom and each moving link has 6 parameters of 3 positions and 3 orientations. In essence, each joint brings with it 5 constraints and so the number of degrees of freedom in the robot arm is the same amount of joints in the robot arm. If there are 6 joints, then there are 6 degrees of freedom. If there are 2 joints, then there are only 2 degrees of freedom.

The professor goes on to define operation coordinates and joint coordinates. Operation coordinates refer to the parameters defining the position and orientation of the end-effector. These parameters are a combination of Euler angles and position vectors that describe the end-effector element. Joint coordinates are the parameters that define the position and orientation of each joint in the system. There are two types of joints, each providing different constraints to a system. A revolute joint limits movement across any axis and allows rotation to only one axis. A prismatic joint limits all rotation on any axis and allows movement across only one axis.

An interesting phenomenon that occurs with multi-jointed robots is redundancy. A robot is said to be redundant if the number of degrees of freedom of the robot is greater than the number of degrees of freedom of the end effector. Redundancy is when a robot has an infinite number of ways to make the end effector reach its destination. The degrees of redundancy can be measured by taking the difference between the number of degrees of freedom of the robot minus the number of degrees of freedom of the end effector. Redundancy is important for when a robot reaches an obstacle and needs another way around to reach its destination.

Finally, the last topic of the video is about rotation and transformation matrices to describe a point from differing coordinate frames. The rotation matrix is a 3x3 matrix that describes the orientation of a point with respect to a coordinate frame. A rotation matrix can be used to describe the orientation of the same point with respect to a different coordinate frame. Combine this with the translation vector of the point and you get a 4x4 transformation matrix that can describe the point’s orientation and position with respect to another coordinate frame.

**In-Class Assignment Unit 2**

**Question 1:**

function [T] = tmat(xr,yr, zr, xt, yt, zt)

% This function takes in 3 rotation parameters and the translation

% vector, and outputs a 4x4 transformation matrix follwing the ZYZ

% convention

xr = xr \* pi/180; %converting inputted degrees into radians for MATLAB to process correctly

yr = yr \* pi/180;

zr = zr \* pi/180;

T = [cos(xr) -sin(xr) 0; sin(xr) cos(xr) 0; 0 0 1]; % Making the x rotation matrix if there's any x rotation. Here I am following the ZYZ Convention

T = T \* [cos(yr) 0 sin(yr); 0 1 0; -sin(yr) 0 cos(yr)]; % Multiplying the x rotation matrix with the y matrix to compute any inputted rotations

T = T \* [cos(zr) -sin(zr) 0; sin(zr) cos(zr) 0; 0 0 1]; % Again, Multiplying the x and y rotation matrix with the z matrix to compute any inputted rotations

T = [T [xt yt zt]'; 0 0 0 1]; % Forming the transformation matrix

end

>> tmat (0,0,90, 10,0,0)

ans =

0.0000 -1.0000 0 10.0000

1.0000 0.0000 0 0

0 0 1.0000 0

0 0 0 1.0000

>> trotz(90, 'deg')

ans =

0 -1 0 0

1 0 0 0

0 0 1 0

0 0 0 1

>> ans(1,4) = 10

ans =

0 -1 0 10

1 0 0 0

0 0 1 0

0 0 0 1

>>

**Question 2:**

>> BTA = [0.866 -0.5 0 4; 0.5 0.866 0 3; 0 0 1 0; 0 0 0 1]

BTA =

0.8660 -0.5000 0 4.0000

0.5000 0.8660 0 3.0000

0 0 1.0000 0

0 0 0 1.0000

>> BRA = [ 0.8660 -0.5000 0;

0.5000 0.8660 0;

0 0 1.0000]

BRA =

0.8660 -0.5000 0

0.5000 0.8660 0

0 0 1.0000

>> ARB = BRA'

ARB =

0.8660 0.5000 0

-0.5000 0.8660 0

0 0 1.0000

>> Porg = [4;3;0]

Porg =

4

3

0

>> PA = -ARB\*Porg

PA =

-4.9640

-0.5980

0

>> ATB = [ARB PA; 0 0 0 1]

ATB =

0.8660 0.5000 0 -4.9640

-0.5000 0.8660 0 -0.5980

0 0 1.0000 0

0 0 0 1.0000

**Question 3:**

function [T] = tmat(xr,yr, zr, xt, yt, zt)

% This function takes in 3 rotation parameters and the translation

% vector, and outputs a 4x4 transformation matrix follwing the ZYZ

% convention

xr = xr \* pi/180; %converting inputted degrees into radians for MATLAB to process correctly

yr = yr \* pi/180;

zr = zr \* pi/180;

T = [cos(xr) -sin(xr) 0; sin(xr) cos(xr) 0; 0 0 1]; % Making the x rotation matrix if there's any x rotation. Here I am following the ZYZ Convention

T = T \* [cos(yr) 0 sin(yr); 0 1 0; -sin(yr) 0 cos(yr)]; % Multiplying the x rotation matrix with the y matrix to compute any inputted rotations

T = T \* [cos(zr) -sin(zr) 0; sin(zr) cos(zr) 0; 0 0 1]; % Again, Multiplying the x and y rotation matrix with the z matrix to compute any inputted rotations

T = [T [xt yt zt]'; 0 0 0 1]; % Forming the transformation matrix

end

>> eul2tr(90,45,90,'deg')

ans =

-1.0000 0 0 0

0 -0.7071 0.7071 0

0 0.7071 0.7071 0

0 0 0 1.0000

>> tmat(90,45,90,0,0,0)

ans =

-1.0000 -0.0000 0.0000 0

0.0000 -0.7071 0.7071 0

-0.0000 0.7071 0.7071 0

0 0 0 1.0000

>> tmat(52,46,93,0,0,0)

ans =

-0.8093 -0.3858 0.4429 0

0.5862 -0.5789 0.5668 0

0.0376 0.7184 0.6947 0

0 0 0 1.0000

>> eul2tr(52,46,93,'deg')

ans =

-0.8093 -0.3858 0.4429 0

0.5862 -0.5789 0.5668 0

0.0376 0.7184 0.6947 0

0 0 0 1.0000

**Question 4**

**1.1:** >> BRA = rotz(30, 'deg')

BRA =

0.8660 -0.5000 0

0.5000 0.8660 0

1. 0 1.0000

**1.2:**

>> BTA = tmat(0,0,30,0,0,0)

BTA =

0.8660 -0.5000 0 0

0.5000 0.8660 0 0

0 0 1.0000 0

0 0 0 1.0000

>> PB = [0,2,0,1]'

PB =

0

2

0

1

>> PA = BTA \* PB

PA =

-1.0000

1.7321

0

1.0000

**1.3:**

>> BTA = trotz(30,'deg')

BTA =

0.8660 -0.5000 0 0

0.5000 0.8660 0 0

0 0 1.0000 0

0 0 0 1.0000

>> PB = [0,2,0,1]'

PB =

0

2

0

1

>> PA = BTA \* PB

PA =

-1.0000

1.7321

0

1.0000

**1.4:**

>> BTA = tmat(0,0,32,0,0,0)

BTA =

0.8480 -0.5299 0 0

0.5299 0.8480 0 0

0 0 1.0000 0

0 0 0 1.0000

>> BTA = trotz(32,'deg')

BTA =

0.8480 -0.5299 0 0

0.5299 0.8480 0 0

0 0 1.0000 0

0 0 0 1.0000

>> PB = [0,2,0,1]'

PB =

0

2

0

1

>> PA = BTA \* PB

PA =

-1.0598

1.6961

0

1.0000

**Question 5:**

**2.1:**

>> BTA = tmat(0,0,30,10,5,0)

BTA =

0.8660 -0.5000 0 10.0000

0.5000 0.8660 0 5.0000

0 0 1.0000 0

0 0 0 1.0000

**2.2:**

>> BTA = tmat(0,0,30,10,5,0)

BTA =

0.8660 -0.5000 0 10.0000

0.5000 0.8660 0 5.0000

0 0 1.0000 0

0 0 0 1.0000

>> PB = [3;7;0;1]

PB =

3

7

0

1

>> PA = BTA \* PB

PA =

9.0981

12.5622

0

1.0000

**2.3**

>> BTA = tmat(0,0,25,15,9,2)

BTA =

0.9063 -0.4226 0 15.0000

0.4226 0.9063 0 9.0000

0 0 1.0000 2.0000

0 0 0 1.0000

>> PB = [3;7;0;1]

PB =

3

7

0

1

>> PA = BTA \* PB

PA =

14.7606

16.6120

2.0000

1.0000

>> BTA = trotz(25,'deg')

BTA =

0.9063 -0.4226 0 0

0.4226 0.9063 0 0

0 0 1.0000 0

0 0 0 1.0000

>> BTA(1,4) = 15

BTA =

0.9063 -0.4226 0 15.0000

0.4226 0.9063 0 0

0 0 1.0000 0

0 0 0 1.0000

>> BTA(2,4) = 9

BTA =

0.9063 -0.4226 0 15.0000

0.4226 0.9063 0 9.0000

0 0 1.0000 0

0 0 0 1.0000

>> BTA(3,4) = 2

BTA =

0.9063 -0.4226 0 15.0000

0.4226 0.9063 0 9.0000

0 0 1.0000 2.0000

0 0 0 1.0000

>> PB = [3;7;0;1]

PB =

3

7

0

1

>> PA = BTA \* PB

PA =

14.7606

16.6120

2.0000

1.0000

**Video Journal 3**

Video: Lecture 3 | Intro to Robotics

Key Terms: **Spatial Descriptions:** Ways of describing an object in space

**Transformation Matrices:** 4x4 matrices capable of defining position and orientation of an object

**Representations:** The technique used to define coordinates on an axis

This lecture is a more thorough deep dive into the topic of spatial descriptions. The first half of the lecture is all about transformation matrices. A transformation matrix is a homogeneous 4x4 matrix that can be interpreted in several ways and is used to describe position and orientation. The first way to use transformation matrices is to describe a coordinate frame ‘B’ relative to another frame ‘A’ and the origin point of ‘B’ with respect to ‘A’. Another way to use these 4x4 matrices is to convert a vector or a point with respect to coordinate frame ‘A’ into a vector or point with respect to frame ‘B’. The vector or point does not move from its location in space, rather the perspective from which these vectors or points are described from is changed. This action is known as **Transform mapping.** Lastly, Transformation matrices can be used as an operator to change a vector after a translation.

Transformation matrices can be used to find other transformation matrices simply by multiplying matrices together. Similarly, the transformation matrix for going in the opposite direction is just the inverse of the matrix, although taking the inverse of a transformation matrix is a little bit different than usual. For example, if the transformation matrix from A to B is known, the transformation matrix from B to A is just the inverse of the transformation matrix from A to B. This is necessary and critical for the function of robot whose position and orientation is always changing. By having the transformation matrices for every joint in a robot, it becomes possible to calculate and predict where and how a robot should move.

The latter half of the lecture moves onto the topic of representations. There are several ways to describe the end-effector in space, and each way has it advantages. Most commonly used and understood method to represent position is through cartesian coordinates. However, cartesian does have its downsides. Other representations of positions can be cylindrical or sphereical coordinates. One case where cartesian wouldn’t be as advantageous as cylindrical for example would be when a vector is aligned with an axis and is spinning along that axis. Cylindrical coordinates would be better fit to represent that spinning motion.

Rotation representation also has a couple ways to be represented. The two forms of rotation representations are Z-Y-X Euler angles or X-Y-Z fixed angles (yaw, pitch, roll). Both representations can be used to find the exact same rotation, just in a different way. Fixed angles are much more intuitive and is used in aviation for small angle adjustments while flying. There is a problem with rotation representation however, and it’s when two axes align and cannot be distinguished between each other. This causes a singularity that must be dealt with for the robot to function optimally.

**In-Class Assignment Unit 3**

2.

1. The DH parameters for the puma560 robot are ‘d’, ‘a’, and ‘alpha’ which are link length, link offset, and link twist respectively. For each individual joint they are:
   1. L(1) = Revolute('d', 0,'a', 0, alpha', pi/2
   2. L(2) = Revolute('d', 0, 'a', 0.4318, 'alpha', 0,
   3. L(3) = Revolute('d', 0.15005, 'a', 0.0203, 'alpha', -pi/2
   4. L(4) = Revolute('d', 0.4318, 'a', 0, 'alpha', pi/2, ...
   5. L(5) = Revolute('d', 0, 'a', 0, 'alpha', -pi/2, ...
   6. L(6) = Revolute('d', 0, 'a', 0, 'alpha', 0, ...
2. The revolute function is manually filled out and assigned to a link in the chain using the L() command. Doing this multiple times with varying DH parameters creates a manipulatable robot arm.
3. Some other parameters used in the Revolute command are ‘jointtype’, ‘mdh’, ‘offset’, ‘name’ and ‘flip’

3.

A. Chart, radar chart

Description automatically generated qr pose

B. Chart, radar chart

Description automatically generated qz pose

Chart

Description automatically generated qs pose

Chart, radar chart

Description automatically generated qn pose

C. Chart, radar chart

Description automatically generated Random Angles

**Video Journal 4**

Video: Lecture 4 | Intro to Robotics

Key Terms: **Denavit – Hartenburg Notation:** A set of parameters that describe a link of a robot and how that link can move in space

**Workspace:** The area where the robot can move around in

The lecture starts off talking about link descriptions or how to describe a link in a robot arm. To do this, we must find the relationships between links. One of these are the mutual perpendicular between link axis, or link length, denoted by ‘a’. Another relationship is the rotation along an axis denoted by alpha, a.k.a link twist. These two variables, once defined, are always constant.

Now we get to the last two parameters which change between a constant or a variable depending on the type of joint being used. First, we have the link offset, or ‘d’, which is variable in a prismatic joint but constant in a revolute joint. Lastly, we have the joint angle, denoted by theta, which is variable in a revolute joint but constant in a prismatic joint. With these four parameters, we have everything we need to go from one coordinate frame to the next. Once at the new frame, we need another 4 parameters to describe moving from the new frame, to another new frame. These 4 parameters are known as Denavit-Hartenburg parameters, or DH parameters for short. There are many variations of the DH notation but DH notation is the base from which these variations are derived from.

The lecture goes on to talk about how to attach frames after defining them through the DH parameters. There are certain steps to do this. First, you have to find the normal between the frames. From there, you can take the intersections as origins a define your z-axis along the frame axis and the x-axis along the common normal.

In the case of intersecting joint angles, to attach the frames together, you find the perpendicular between the two z-axis and you obtain the x-axis. Here, you can choose which direction will be the x-axis and the corresponding alpha angle will change depending on your decision.