

Robot Arms, Hands: Kinematics





Kinematics studies the motion of bodies



What a robot arm and hand can do



• Martin 1992-'97 PhD work

What a robot arm and hand can do



• Camilo 2011-? PhD work



Robotics field

- 6 Million mobile robots
 - From \$100 roomba to \$millions Mars rovers
- 1 million robot arms
 - Usually \$20,000-100,000, some millions
- Value of industrial robotics: \$25 billion
- Arms crucial for these industries:
 - Automotive (Welding, painting, some assembly)
 - Electronics (Placing tiny components on PCB)
 - General: Pack boxes, move parts from conveyor to machines



An classic arm - The PUMA 560





The PUMA 560 has SIX revolute joints A revolute joint has ONE degree of freedom (1 DOF) that is defined by its angle

An modern arm - The Barrett WAM





- The WAM has **SEVEN** revolute joints.
- Similar motion (Kinematics) to human

UA Robotics Lab platform 2 arm mobile manipulator



- 2 WAM arms, steel cable transmission and drive
- Segway mobile platform
- 2x Quad core computer platform.
- Battery powered, 4h run time.

Robotics challenges



Navigation '05

Manipulation '11-14

Humanoids '12-

Build or buy?

• Off the shelf kits:



Lynxmotion



• Build your own:





Mathematical modeling





Robot

Strategy:

- 1. Model each joint separately
- 2. Combine joints and linkage lengths

http://www.societyofrobots.com/robot_arm_tutorial.shtml

Other basic joints





Spherical Joint 3 DOF (Variables - Y₁, Y₂, Y₃)

Example Matlab robot

Successive translation and rotation



```
% robocop Simulates a 3 joint robot
function Jpos =
    robocop(theta1,theta2,theta3,L1,L2,L3,P0)
Rxy1 = [cos(theta1) sin(theta1) 0
          -sin(theta1) cos(theta1) 0
                                      1];
          0
                               0
Rxz2 = [cos(theta2) \ 0 \ sin(theta2)]
          0
                         1 0
          -sin(theta2) 0 cos(theta2)];
Rxz3 = [cos(theta3) \ 0 \ sin(theta3)]
          0
                          1
                               0
          -sin(theta3) 0 cos(theta3)];
P1 = P0 + Rxy1*[L1 \ 0 \ 0]';
P2 = P1 + Rxy1*Rxz2*[L2 0 0]';
P3 = P2 + Rxy1*Rxz2*Rxz3*[L3 0 0]';
Jpos = [P0 P1 P2 P3];
```

Problem: Lots of coordinate frames to calibrate

Robot

- Base frame
- End-effector frame
- Object



Problem: Lots of coordinate frames to calibrate

Robot

- Base frame
- End-effector frame
- Object

Camera

- Center of projection
- Different models



We are interested in two kinematics topics

Forward Kinematics (angles to position)

What you are given:	The length of each link
	The angle of each joint

What you can find:

The position of any point (i.e. it's (x, y, z) coordinates

Inverse Kinematics (position to angles)

What you are given:	The length of each link		
	The position of some point in the world		
(reachable)			

What you can find:

The angles of each joint needed to obtain that position



(Note : P_x , P_y are relative to the original coordinate frame. Translation followed by rotation is different than rotation followed by translation.)

In other words, knowing the coordinates of a point (V^N, V^O) in some coordinate frame (NO) you can find the position of that point relative to your original coordinate frame (X^0Y^0) .

HOMOGENEOUS REPRESENTATION Putting it all into a Matrix

$$\mathbf{V}^{XY} = \begin{bmatrix} \mathbf{V}^{X} \\ \mathbf{V}^{Y} \end{bmatrix} = \begin{bmatrix} \mathbf{P}_{x} \\ \mathbf{P}_{y} \end{bmatrix} + \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \begin{bmatrix} \mathbf{V}^{N} \\ \mathbf{V}^{O} \end{bmatrix}$$
 What we found by doing a translation and a rotation
$$= \begin{bmatrix} \mathbf{V}^{X} \\ \mathbf{V}^{Y} \\ \mathbf{1} \end{bmatrix} = \begin{bmatrix} \mathbf{P}_{x} \\ \mathbf{P}_{y} \\ \mathbf{1} \end{bmatrix} + \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{V}^{N} \\ \mathbf{V}^{O} \\ \mathbf{1} \end{bmatrix}$$
 Padding with 0's and 1's
$$= \begin{bmatrix} \mathbf{V}^{X} \\ \mathbf{V}^{Y} \\ \mathbf{1} \end{bmatrix} = \begin{bmatrix} \cos \theta & -\sin \theta & \mathbf{P}_{x} \\ \sin \theta & \cos \theta & \mathbf{P}_{y} \\ \mathbf{0} & \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{V}^{N} \\ \mathbf{V}^{O} \\ \mathbf{1} \end{bmatrix}$$
 Simplifying into a matrix form
$$\mathbf{H} = \begin{bmatrix} \cos \theta & -\sin \theta & \mathbf{P}_{x} \\ \sin \theta & \cos \theta & \mathbf{P}_{y} \\ \mathbf{0} & \mathbf{0} & 1 \end{bmatrix}$$
 Homogenous Matrix for a Translation in XY plane, followed by a Rotation around the z-axis



Homogeneous Matrices in 3D



the three.

Homogeneous Continued....



The rotation and translation part can be combined into a single homogeneous matrix IF and ONLY IF both are relative to the same coordinate frame.





Substituting for $\begin{bmatrix} W^{T} \\ W^{J} \\ W^{K} \end{bmatrix}$ $\begin{bmatrix} W^{X} \\ W^{Y} \\ W^{Z} \\ 1 \end{bmatrix} = \begin{bmatrix} i_{x} & j_{x} & k_{x} & T_{x} \\ i_{y} & j_{y} & k_{y} & T_{y} \\ i_{z} & j_{z} & k_{z} & T_{z} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} n_{i} & o_{i} & a_{i} & P_{i} \\ n_{j} & o_{j} & a_{j} & P_{j} \\ n_{k} & o_{k} & a_{k} & P_{k} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} W^{N} \\ W^{O} \\ W^{A} \\ 1 \end{bmatrix}$



Notice that H can also be written as:

$$H = \begin{bmatrix} 1 & 0 & 0 & T_x \\ 0 & 1 & 0 & T_y \\ 0 & 0 & 1 & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} i_x & j_x & k_x & 0 \\ i_y & j_y & k_y & 0 \\ i_z & j_z & k_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & P_i \\ 0 & 1 & 0 & P_j \\ 0 & 0 & 1 & P_k \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} n_i & o_i & a_i & 0 \\ n_j & o_j & a_j & 0 \\ n_k & o_k & a_k & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

H = (Translation relative to the XYZ frame) * (Rotation relative to the XYZ frame)
 * (Translation relative to the IJK frame) * (Rotation relative to the IJK frame)



One more variation on finding H:

H = (Rotate so that the X-axis is aligned with T)

* (Translate along the new t-axis by $\parallel T \parallel$ (magnitude of T))

* (Rotate so that the t-axis is aligned with P)

* (Translate along the p-axis by || P ||)

* (Rotate so that the p-axis is aligned with the O-axis)

This method might seem a bit confusing, but it's actually an easier way to solve our problem given the information we have. Here is an example...

Forward Kinematics



The Situation:

You have a robotic arm that starts out aligned with the x_o -axis. You tell the first link to move by Y_1 and the second link to move by Y_2 .

The Quest:

What is the position of the end of the robotic arm?

Solution:

1. Geometric Approach

This might be the easiest solution for the simple situation. However, notice that the angles are measured relative to the direction of the previous link. (The first link is the exception. The angle is measured relative to it's initial position.) For robots with more links and whose arm extends into 3 dimensions the geometry gets much more tedious.

2. Algebraic Approach

Involves coordinate transformations.

Example Problem:

You are have a three link arm that starts out aligned in the x-axis. Each link has lengths l_1 , l_2 , l_3 , respectively. You tell the first one to move by Y_1 , and so on as the diagram suggests. Find the Homogeneous matrix to get the position of the yellow dot in the X⁰Y⁰ frame.



2

The position of the yellow dot relative to the X^3Y^3 frame is $(l_3, 0)$. Multiplying H by that position vector will give you the coordinates of the yellow point relative the the X^0Y^0 frame.

Slight variation on the last solution:

Make the yellow dot the origin of a new coordinate X^4Y^4 frame



More on Forward Kinematics...

Denavit - Hartenberg Parameters



IDEA: Each joint is assigned a coordinate frame. Using the Denavit-Hartenberg notation, you need 4 parameters to describe how a frame (i) relates to a previous frame (i - 1).

THE PARAMETERS/VARIABLES: α, a, d, Y

The Parameters



You can align the two axis just using the 4 parameters

1) $a_{(i-1)}$ <u>Technical Definition:</u> $a_{(i-1)}$ is the length of the perpendicular between the joint axes. The joint axes is the axes around which revolution takes place which are the $Z_{(i-1)}$ and $Z_{(i)}$ axes. These two axes can be viewed as lines in space. The common perpendicular is the shortest line between the two axis-lines and is perpendicular to both axis-lines.

 $a_{(i-1) cont...}$ <u>Visual Approach</u> - "A way to visualize the link parameter $a_{(i-1)}$ is to imagine an expanding cylinder whose axis is the $Z_{(i-1)}$ axis - when the cylinder just touches the joint axis *i* the radius of the cylinder is equal to $a_{(i-1)}$." (Manipulator Kinematics)

<u>It's Usually on the Diagram Approach</u> - <u>If the diagram already specifies the</u> various coordinate frames, then the common perpendicular is usually the $X_{(i-1)}$ axis. So $a_{(i-1)}$ is just the displacement along the $X_{(i-1)}$ to move from the (i-1) frame to the *i* frame.

If the link is prismatic, then $a_{(i-1)}$ is a variable, not a parameter.



2) $\alpha_{(i-1)}$

<u>Technical Definition</u>: Amount of rotation around the common perpendicular so that the joint axes are parallel.

i.e. How much you have to rotate around the $X_{(i-1)}$ axis so that the $Z_{(i-1)}$ is pointing in the same direction as the Z_i axis. Positive rotation follows the right hand rule.

3) $d_{(i-1)}$ <u>Technical Definition</u>: The displacement along the Z_i axis needed to align the $a_{(i-1)}$ common perpendicular to the a_i common perpendicular.

In other words, displacement along the Z_i to align the $X_{(i-1)}$ and X_i axes.



4) Y_i

Amount of rotation around the Z_i axis needed to align the $X_{(i-1)}$ axis with the X_i axis.

The Denavit-Hartenberg Matrix

$$\begin{bmatrix} \cos \theta_{i} & -\sin \theta_{i} & 0 & a_{(i-1)} \\ \sin \theta_{i} \cos \alpha_{(i-1)} & \cos \theta_{i} \cos \alpha_{(i-1)} & -\sin \alpha_{(i-1)} & -\sin \alpha_{(i-1)} d_{i} \\ \sin \theta_{i} \sin \alpha_{(i-1)} & \cos \theta_{i} \sin \alpha_{(i-1)} & \cos \alpha_{(i-1)} & \cos \alpha_{(i-1)} d_{i} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Just like the Homogeneous Matrix, the Denavit-Hartenberg Matrix is a transformation matrix from one coordinate frame to the next. Using a series of D-H Matrix multiplications and the D-H Parameter table, the final result is a transformation matrix from some frame to yc_{λ}

Put the transformation here





Denavit-Hartenberg Link Parameter Table

Notice that the table has two uses:

1) To describe the robot with its variables and parameters.

2) To describe some state of the robot by having a numerical values for the variables.

i d_i θ_i $a_{(i-1)}$ $\alpha_{(i-1)}$ 0 θ_0 0 0 0 1 0 0 θ_1 a_0 2 θ_2 -90 d_2 a_1



$$\mathbf{V}^{\mathbf{X}_{0}\mathbf{Y}_{0}\mathbf{Z}_{0}} = \mathbf{T}\begin{bmatrix} \mathbf{V}^{\mathbf{X}_{2}} \\ \mathbf{V}^{\mathbf{Y}_{2}} \\ \mathbf{V}^{\mathbf{Z}_{2}} \\ \mathbf{1} \end{bmatrix}$$

$$\mathbf{T} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \begin{pmatrix} 1 \\ 2 \end{pmatrix}$$

Note: T is the D-H matrix with $(i-1) = 0$ and $i = 1$.

i	Q _(i-1)	a _(i-1)	d_i	$\mathbf{\Theta}_i$
0	0	0	0	θ ₀
1	0	a ₀	0	θ_1
2	-90	a ₁	d ₂	θ_2

$${}_{1}^{0}T = \begin{bmatrix} \cos\theta_{1} & -\sin\theta_{1} & 0 & a_{0} \\ \sin\theta_{1} & \cos\theta_{1} & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

This is a translation by a_0 followed by a rotation around the Z_1 axis

$${}_{0}\mathbf{T} = \begin{bmatrix} \mathbf{cos}\theta_{0} & -\mathbf{sin}\theta_{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{sin}\theta_{0} & \mathbf{cos}\theta_{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \end{bmatrix}$$

This is just a rotation around the Z_0 axis

$${}^{1}_{2}T = \begin{bmatrix} \cos\theta_{2} & -\sin\theta_{2} & 0 & a_{1} \\ 0 & 0 & 1 & d_{2} \\ -\sin\theta_{2} & -\cos\theta_{2} & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

This is a translation by a_1 and then d_2 followed by a rotation around the X_2 and Z_2 axis

 $\mathbf{T} = ({}_{0}\mathbf{T})({}_{1}^{0}\mathbf{T})({}_{2}^{1}\mathbf{T})$

Inverse Kinematics

From Position to Angles

A Simple Example

Revolute and Prismatic Joints Combined

Y

 (\mathbf{x},\mathbf{y})

S

►X



More Specifically:

$$\theta = \arctan 2(\frac{y}{x})$$

arctan2() specifies that it's in the first quadrant

Finding S:

$$S = \sqrt{(x^2 + y^2)}$$

Inverse Kinematics of a Two Link Manipulator



Given: l_1, l_2, x, y

Find: Y_1, Y_2

Redundancy:

A unique solution to this problem does not exist. Notice, that using the "givens" two solutions are possible. Sometimes no solution is possible.



The Geometric Solution



Using the Law of Cosines:

$$\frac{\sin B}{b} = \frac{\sin C}{c}$$
$$\frac{\sin \overline{\theta}_1}{l_2} = \frac{\sin(180 - \theta_2)}{\sqrt{x^2 + y^2}} = \frac{\sin(\theta_2)}{\sqrt{x^2 + y^2}}$$
$$\theta_1 = \overline{\theta}_1 + \alpha$$
$$\alpha = \arctan 2 \left(\frac{y}{x}\right)$$

Using the Law of Cosines:

$$c^{2} = a^{2} + b^{2} - 2ab \cos C$$

$$(x^{2} + y^{2}) = l_{1}^{2} + l_{2}^{2} - 2l_{1}l_{2} \cos(180 - \theta_{2})$$

$$\cos(180 - \theta_{2}) = -\cos(\theta_{2})$$

$$\cos(\theta_{2}) = \frac{x^{2} + y^{2} - l_{1}^{2} - l_{2}^{2}}{2l_{1}l_{2}}$$

$$\theta_{2} = \arccos\left(\frac{x^{2} + y^{2} - l_{1}^{2} - l_{2}^{2}}{2l_{1}l_{2}}\right)$$

Redundant since θ_2 could be in the first or fourth quadrant.

Redundancy caused since θ_2 has two possible values

$$\theta_1 = \arcsin\left(\frac{l_2\sin(\theta_2)}{\sqrt{x^2 + y^2}}\right) + \arctan 2\left(\frac{y}{x}\right)$$

The Algebraic Solution



$$c_{1} = \cos \theta_{1}$$

$$c_{1+2} = \cos(\theta_{2} + \theta_{1})$$
(1) $x = l_{1} c_{1} + l_{2} c_{1+2}$
(2) $y = l_{1} s_{1} + l_{2} \sin_{1+2}$
(3) $\theta = \theta_{1} + \theta_{2}$

$$(1)^{2} + (2)^{2} = x^{2} + y^{2} =$$

$$= \left(l_{1}^{2} c_{1}^{2} + l_{2}^{2} (c_{1+2})^{2} + 2l_{1}l_{2} c_{1}(c_{1+2}) \right) + \left(l_{1}^{2} s_{1}^{2} + l_{2}^{2} (\sin_{1+2})^{2} + 2l_{1}l_{2} s_{1}(\sin_{1+2}) \right)$$

$$= l_{1}^{2} + l_{2}^{2} + 2l_{1}l_{2} (c_{1}(c_{1+2}) + s_{1}(\sin_{1+2}))$$

$$= l_{1}^{2} + l_{2}^{2} + 2l_{1}l_{2} c_{2}$$
Only Unknown
$$Note:$$

$$\cos(a_{-}^{+}b) = (\cos a)(\cos b)_{+}^{-}(\sin a)(\sin b)$$

$$\sin(a_{-}^{+}b) = (\cos a)(\sin b)_{-}^{+}(\cos b)(\sin a)$$

The Numeric solution

> Now given desired Cartesian position [X,Y,Z] solve numerically for the corresponding joint angles $[r_1r_2r_3]$:

$$0 = \begin{bmatrix} \mathbf{X} \\ \mathbf{Y} \\ \mathbf{Z} \\ \mathbf{1} \end{bmatrix} - f(r)$$

The Numeric solution: How to solve? Newton's Metod

Function: $W^4 = f(r) = \mathbf{H}(r, l)I$

3

Jacobian J = matrix of partial derivatives:

 X^4



 Y^2

 $\mathbf{J} = \left[\frac{\partial f_i(r)}{\partial r_j}\right]$

Newton's method:

Guess initial joint angles r

Iterate

```
J^*dr = W-f(r)
```

r = r + dr

If guess is close enough r converges to solution.

Otherwise may diverge.

Newton's Metod: Convergence issues



Use a start position with known W and r (e.g. Wi = $[0, l_1 + l_2 + l_3, 0, 1]^T$ for r = 0)

Let next Wk close to this initial. Use r0 as initial guess for r1 Iterate

$$J^*dr = Wk - f(r)$$

$$r = r + dr$$

r guess is close so r converges to solution.

Newton's Metod: Convergence issues



To make a large movement, divide the total distance from (known) initial Wi to the new final Wf into small steps • Wk e.g. on a line

•Try this in lab!

Resolved rate control

- Here instead of computing an inverse kinematics solution then move the robot to that point, we actually move the robot dr for every iteration in newtons method.
- Let $dr \rightarrow 0$, then we can view this as velocity control:

 $\dot{r} = \mathbf{J}(r(t))^{-1} \dot{w}$

 $\dot{w} = v = Cartesian translation velovity$

Conclusion

- Forward kinematics can be tedious for multilink arms
- Inverse kinematics can be solved algebraically or numerically. The latter is more common for complex arms or vision-guided control (later)
- Limitations: We avoided details of the various angular representations (Euler, quarternion or exponentials) and their detailed use in Kinematics. (this typically takes



Quick Math Review



Unit Vector

Vector in the direction of a chosen vector but whose magnitude is 1.



Quick Matrix Review

Matrix Multiplication:

An $(m \ge n)$ matrix A and an $(n \ge p)$ matrix B, can be multiplied since the number of columns of A is equal to the number of rows of B.

Non-Commutative Multiplication AB is NOT equal to BA

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} * \begin{bmatrix} e & f \\ g & h \end{bmatrix} = \begin{bmatrix} (ae+bg) & (af+bh) \\ (ce+dg) & (cf+dh) \end{bmatrix}$$

Matrix Addition:

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} + \begin{bmatrix} e & f \\ g & h \end{bmatrix} = \begin{bmatrix} (a+e) & (b+f) \\ (c+g) & (d+h) \end{bmatrix}$$

Basic Transformations Moving Between Coordinate Frames

Translation Along the X-Axis



 P_x = distance between the XY and NO coordinate planes

Notation:
$$\overline{\mathbf{V}}^{\mathbf{X}\mathbf{Y}} = \begin{bmatrix} \mathbf{V}^{\mathbf{X}} \\ \mathbf{V}^{\mathbf{Y}} \end{bmatrix} \qquad \overline{\mathbf{V}}^{\mathbf{NO}} = \begin{bmatrix} \mathbf{V}^{\mathbf{N}} \\ \mathbf{V}^{\mathbf{O}} \end{bmatrix} \qquad \overline{\mathbf{P}} = \begin{bmatrix} \mathbf{P}_{\mathbf{x}} \\ \mathbf{0} \end{bmatrix}$$

Writing $\overline{\mathbf{V}}^{\mathbf{X}\mathbf{Y}}$ in terms of $\overline{\mathbf{V}}^{\mathbf{NO}}$



$$\overline{\mathbf{V}}^{\mathbf{X}\mathbf{Y}} = \begin{bmatrix} \mathbf{P}_{\mathbf{X}} + \mathbf{V}^{\mathbf{N}} \\ \mathbf{V}^{\mathbf{O}} \end{bmatrix} = \overline{\mathbf{P}} + \overline{\mathbf{V}}^{\mathbf{NO}}$$



$$\overline{\mathbf{V}}^{\mathbf{X}\mathbf{Y}} = \overline{\mathbf{P}} + \overline{\mathbf{V}}^{\mathbf{NO}} = \begin{bmatrix} \mathbf{P}_{\mathbf{X}} + \mathbf{V}^{\mathbf{N}} \\ \mathbf{P}_{\mathbf{Y}} + \mathbf{V}^{\mathbf{O}} \end{bmatrix}$$

Using Basis Vectors

Basis vectors are unit vectors that point along a coordinate axis



$$\overline{\mathbf{V}}^{\mathbf{NO}} = \begin{bmatrix} \mathbf{V}^{\mathbf{N}} \\ \mathbf{V}^{\mathbf{O}} \end{bmatrix} = \begin{bmatrix} \|\mathbf{V}^{\mathbf{NO}}\|\mathbf{cos}\theta \\ \|\mathbf{V}^{\mathbf{NO}}\|\mathbf{sin}\theta \end{bmatrix} = \begin{bmatrix} \|\mathbf{V}^{\mathbf{NO}}\|\mathbf{cos}\theta \\ \|\mathbf{V}^{\mathbf{NO}}\|\mathbf{cos}(90-\theta) \end{bmatrix} = \begin{bmatrix} \overline{\mathbf{V}}^{\mathbf{NO}} \bullet \overline{\mathbf{n}} \\ \overline{\mathbf{V}}^{\mathbf{NO}} \bullet \overline{\mathbf{o}} \end{bmatrix}$$



Y = Angle of rotation between the XY and NO coordinate axis

$$\overline{\mathbf{V}}^{\mathbf{X}\mathbf{Y}} = \begin{bmatrix} \mathbf{V}^{\mathbf{X}} \\ \mathbf{V}^{\mathbf{Y}} \end{bmatrix} \qquad \overline{\mathbf{V}}^{\mathbf{NO}} = \begin{bmatrix} \mathbf{V}^{\mathbf{N}} \\ \mathbf{V}^{\mathbf{O}} \end{bmatrix}$$



 \mathbf{X} Unit vector along X-Axis

 $\underline{\mathbf{\Lambda}}$ Can be considered with respect to the XY coordinates or NO coordinates

$$\left\| \overline{\mathbf{V}}^{\mathbf{X}\mathbf{Y}} \right\| = \left\| \overline{\mathbf{V}}^{\mathbf{NO}} \right\|$$

$$\mathbf{V}^{\mathbf{X}} = \|\overline{\mathbf{V}}^{\mathbf{X}\mathbf{Y}}\| \mathbf{cos} \ \boldsymbol{\alpha} = \|\overline{\mathbf{V}}^{\mathbf{NO}}\| \mathbf{cos} \ \boldsymbol{\alpha} = \overline{\mathbf{V}}^{\mathbf{NO}} \bullet \overline{\mathbf{x}}$$

$$\mathbf{V}^{\mathbf{X}} = (\mathbf{V}^{\mathbf{N}} * \overline{\mathbf{n}} + \mathbf{V}^{\mathbf{O}} * \overline{\mathbf{o}}) \bullet \overline{\mathbf{x}} \qquad \text{(Substituting for V^{NO} using the N and O components of the vector)}$$

$$\mathbf{V}^{\mathbf{X}} = \mathbf{V}^{\mathbf{N}} (\overline{\mathbf{x}} \bullet \overline{\mathbf{n}}) + \mathbf{V}^{\mathbf{O}} (\overline{\mathbf{x}} \bullet \overline{\mathbf{o}})$$

$$= \mathbf{V}^{\mathbf{N}} (\mathbf{cos} \ \boldsymbol{\theta}) + \mathbf{V}^{\mathbf{O}} (\mathbf{cos} (\ \boldsymbol{\theta} + \mathbf{90}))$$

$$= \mathbf{V}^{\mathbf{N}} (\cos \theta) - \mathbf{V}^{\mathbf{O}} (\sin \theta)$$

Similarly....

$$V^{Y} = \|\overline{V}^{NO}\| \sin \alpha = \|\overline{V}^{NO}\| \cos(90 - \alpha) = \overline{V}^{NO} \cdot \overline{y}$$

$$V^{Y} = (V^{N} * \overline{n} + V^{O} * \overline{o}) \cdot \overline{y}$$

$$V^{Y} = V^{N}(\overline{y} \cdot \overline{n}) + V^{O}(\overline{y} \cdot \overline{o})$$

$$= V^{N}(\cos(90 - \theta)) + V^{O}(\cos \theta)$$

$$= V^{N}(\sin \theta) + V^{O}(\cos \theta)$$

So....

$$\mathbf{V}^{\mathbf{X}} = \mathbf{V}^{\mathbf{N}} (\cos \theta) - \mathbf{V}^{\mathbf{O}} (\sin \theta)$$
$$\mathbf{V}^{\mathbf{Y}} = \mathbf{V}^{\mathbf{N}} (\sin \theta) + \mathbf{V}^{\mathbf{O}} (\cos \theta)$$

$$\overline{\mathbf{V}}^{\mathbf{X}\mathbf{Y}} = \begin{bmatrix} \mathbf{V}^{\mathbf{X}} \\ \mathbf{V}^{\mathbf{Y}} \end{bmatrix}$$

Written in Matrix Form

$$\overline{\mathbf{V}}^{\mathbf{X}\mathbf{Y}} = \begin{bmatrix} \mathbf{V}^{\mathbf{X}} \\ \mathbf{V}^{\mathbf{Y}} \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} \mathbf{V}^{\mathbf{N}} \\ \mathbf{V}^{\mathbf{O}} \end{bmatrix}$$

Rotation Matrix about the z-axis

$$y = l_1 s_1 + l_2 \sin_{1+2}$$

= $l_1 s_1 + l_2 s_1 c_2 + l_2 s_2 c_1$
= $c_1 (l_2 s_2) + s_1 (l_1 + l_2 c_2)$

С

S

$$c_{1} = \frac{x + s_{1}(l_{2}s_{2})}{(l_{1} + l_{2}c_{2})}$$

$$y = \frac{x + s_{1}(l_{2}s_{2})}{(l_{1} + l_{2}c_{2})}(l_{2}s_{2}) + s_{1}(l_{1} + l_{2}c_{2})$$
Substituting for c_{1} and simplifying many times
$$= \frac{1}{(l_{1} + l_{2}c_{2})}\left(x \ l_{2}s_{2} + s_{1}(l_{1}^{2} + l_{2}^{2} + 2l_{1}l_{2}c_{2})\right)$$
Notice this is the law of cosines and can be replaced by $x^{2} + y^{2}$

$$s_{1} = \frac{y(l_{1} + l_{2}c_{2}) - x \ l_{2}s_{2}}{x^{2} + y^{2}}$$

$$\theta_{1} = \arcsin\left(\frac{y(l_{1} + l_{2}c_{2}) - x \ l_{2}s_{2}}{x^{2} + y^{2}}\right)$$

Note: $\cos(a_{-}^{+}b) = (\cos a)(\cos b)_{+}^{-}(\sin a)(\sin b)$ $\sin(a_{-}^{+}b) = (\cos a)(\sin b)_{-}^{+}(\cos b)(\sin a)$

We know what θ_2 is from the previous slide. We need to solve for θ_1 . Now we have two equations and two unknowns (sin θ_1 and cos θ_1)

olifying