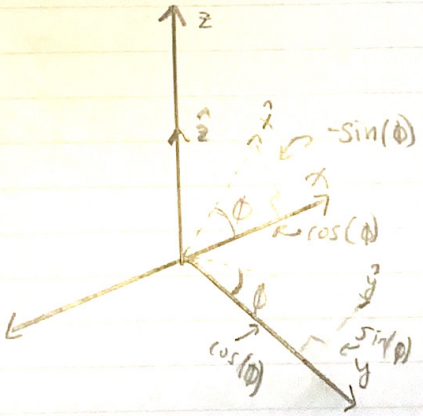
Dean Meluban
Comp 417 - A1
260764246

① 3x3 "Rotation Matrix" $R(\phi)$ that represents change in x, y position after rotating ϕ radians about the ICC is:

$$R(\phi) = \begin{bmatrix} \cos(\phi) & -\sin(\phi) & 0 \\ \sin(\phi) & \cos(\phi) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

To derive R, consider the x, y, z plane and a rotation about the plane (denoted by $\hat{x}, \hat{y}, \hat{z}$) When rotating in this frame of reference, we rotate along the z-axis.



Assume starting position is along x, y, z axis, and rotations are $\hat{x}, \hat{y}, \hat{z}$.

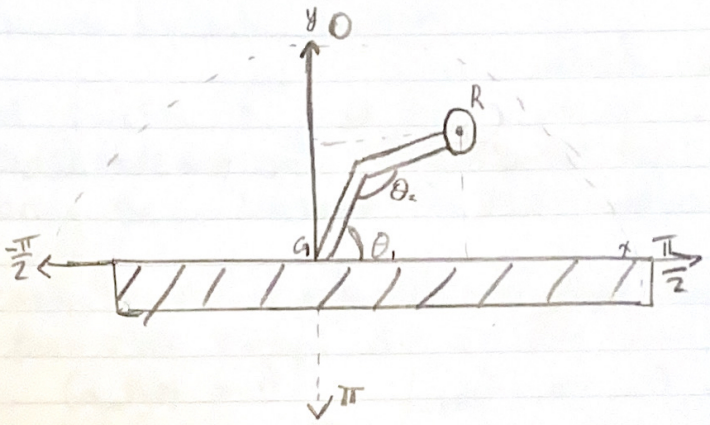
To calculate R, we simply find the projection of x on $\hat{x}$, y on $\hat{y}$ and z on $\hat{z}$. Assume length of all vectors is equal to 1.

Hence, R is essentially the projection of

$$R = \begin{array}{c} \text{ON} \rightarrow \\ \\ x \\ y \\ z \end{array} \begin{array}{c} \begin{array}{ccc} \hat{x} & \hat{y} & \hat{z} \end{array} \\ \begin{bmatrix} - & - & - \\ - & - & - \\ - & - & - \end{bmatrix} \end{array}$$

. From the graph above, finding the projections of $\hat{x}, \hat{y}$, and $\hat{z}$ is found easily using trig. Hence R is defined as above.

②



Limitations: $-\frac{\pi}{2} \leq \theta_1 \leq \frac{\pi}{2}$

$-\pi \leq \theta_2 \leq \pi$

- Frame is called G.
- axes called x, y.
- End effector is called R.

a) state space of the end effector (state = [position, orientation]

$$X = [G_{Px}, G_{Py}, G_{\theta_1}, G_{\theta_2}]$$

- Position of x and y w/ respect to frame G
- Orientation of θ_1, θ_2 w/ respect to frame G.

b) action space of end effector

$$U = [R_{w\theta_1}, R_{w\theta_2}]$$

- Joint turning rates determine velocity of end effector of respective joints. W is Omega for rotational movement.

c) Forward Kinematics function.

- Let function f input two parameters, ① original state, ② command of motion, and output state after command.

An original state will have the state vector of form:

$$X = [G_{px}, G_{py}, G_{\theta_1}, G_{\theta_2}] \quad \text{(from part a)}$$

and a command would be of the form

$$Q = [\Delta\theta_1, \Delta\theta_2] \quad \text{where } \Delta\theta_1, \Delta\theta_2 \text{ are the changes to the joints we want to apply. Note that } \Delta\theta_1, \Delta\theta_2 \text{ must abide by constraints given in problem outline, else command will not execute.}$$

Hence,

```
f(x, q) {
    (check if q is allowed following constraints. If not, return false.
    Let x2 be new state matrix;
    x2 = [Gpx', Gpy', Gθ1 + Δθ1, Gθ2 + Δθ2]
    return x2
}
```

Essentially, update the state by applying a command vector onto the change. G_{px}', G_{py}' are the new position states of the end effector.

d) Inverse Kinematics Function

- Let function f^{-1} input two parameters, ① original state, and ② desired end state. f^{-1} will output the command that will bring the end effector to the desired state from the original state.

Both state vectors are of the form

$x = [G_{px}, G_{py}, G_{\theta_1}, G_{\theta_2}]$ (original)

$\hat{x} = [G_{px'}, G_{py'}, G_{\theta_1'}, G_{\theta_2'}]$ (desired)

- A command will be in the form

$Q = [\Delta_{\theta_1}, \Delta_{\theta_2}]$ where $\Delta_{\theta_1}, \Delta_{\theta_2}$ follow the constraints of the problem outline. If they do not obey constraints, or the desired output is impossible, return false.

Hence,

$f^{-1}(x, \hat{x})$ {
Determine if $\hat{x}$ is permitted by constraints. If not, return false.
Let Q = command matrix
$Q = [G_{\theta_1'} - G_{\theta_1}, G_{\theta_2'} - G_{\theta_2}]$
return Q
}

③ Preliminary Assumptions:

- The robot can only move in 8 cardinal directions
  - ↳ (N, S, E, W, NW, NE, SW, SE)
- The path outlined is the shortest path from start → end
- There are a finite number of edges and vertices, each edge is weighted w/ an integer signifying distance.

a) Bug #1 Algorithm (w/ right hand rule)

No! Consider the first turn of the robot. At that point, there is no obstacle. The Bug Algo says to continue straight until confronted with an obstacle. Hence the Bug Algo could not have generated the path.

b) Bug #2 Algorithm (w/ right hand rule)

No! Same intuition as (a). Algo implies straight line until obstacle. In path defined, robot turns before an obstacle.

c) Dijkstra's Algo

Yes! Dijkstra's, as a shortest path algo, analyzes all possible paths and computes the shortest distance to each vertex. Considering the assumption that vertices & edges are finite, Dijkstra's terminates and could yield the path.

d) A*

Yes! A* is similar to Dijkstra's, just with a heuristic function added to the path estimation computation. Due to the nature of the path & the problem assumptions, and because of (c) being able to find the path, A* will work as well.

c) RRT

Yes! RRT algorithm would be able to generate the path. This is due to the random nature of the RRT. As the tree grows throughout the search space, over many iterations, the probability to generate this path grows.

RRT constructs a tree incrementally from samples over the search space Because we assume # of vertices is finite, RRT will eventually generate the path.