GRENOBLE INP Ense3 UGA

Visual Behavior

Robotics Motion Planning and Control in the Wild through Camera Neural Perception

Theara SENG

Supervised by:

Thibault NEVEU

02 September 2022

Outline

1. Introduction

About the company

Visual Behavior

- Provides scalable solutions for autonomous robots.
- Develops an Artificial Visual Cortex software emulating the human brain to provide robots with a high-level understanding of their surroundings

alobot

Introduction

Mobile Robot Use Cases: Industrial Transport, Logistics, Food Serving, and Autonomous Security Robot.

Critical Challenge:

Solution:

- Localization which uses sensors to estimate the position of the robot.
- Map which represents the physical environment.

2. Trajectory Tracking of a Quadcopter

State Space Representation

$$\begin{cases} \dot{X} = AX + BU \\ Y = CX + DU \end{cases}$$

$$X = [\phi \quad \theta \quad \psi \quad p \quad q \quad r \quad u \quad v \quad w \quad x \quad y \quad z]$$
$$\dot{X} = [\dot{\phi} \quad \dot{\theta} \quad \dot{\psi} \quad \dot{p} \quad \dot{q} \quad \dot{r} \quad \dot{u} \quad \dot{v} \quad \dot{w} \quad \dot{x} \quad \dot{y} \quad \dot{z}]$$

$[x \quad y \quad z \quad \phi \quad \theta \quad \psi]$ containing the linear and angular position of the quadcopter in earth frame.

$[u \quad v \quad w \quad p \quad q \quad r]$ containing the linear and angular velocity of the quadcopter in body frame.

Quadcopter Dynamic Model

$$
\begin{aligned}
v &= Rv_b \quad \text{where} \quad v = [\dot{x} \quad \dot{y} \quad \dot{z}] \quad \text{and} \quad v_b = [u \quad v \quad w] \\
\omega &= T\omega_b \quad \text{where} \quad \omega = [\dot{\phi} \quad \dot{\theta} \quad \dot{\psi}] \quad \text{and} \quad \omega_b = [p \quad q \quad r]
\end{aligned}
$$

$$
\begin{aligned}
\dot{x} &= w[s(\phi)s(\psi) + c(\phi)c(\psi)s(\theta)] - v[c(\phi)s(\psi) - c(\psi)s(\phi)s(\theta)] + u[c(\psi)c(\theta)] \\
\dot{y} &= -w[c(\psi)s(\phi) - c(\phi)s(\psi)s(\theta)] + v[c(\phi)c(\psi) + s(\phi)s(\psi)s(\theta)] + u[c(\theta)s(\psi)] \\
\dot{z} &= w[c(\phi)c(\theta)] + v[c(\theta)s(\phi)] - us(\theta) \\
\dot{\phi} &= p + r[c(\phi)t(\theta)] + q[s(\phi)t(\theta)] \\
\dot{\theta} &= qc(\phi) - rs(\phi) \\
\dot{\psi} &= r\frac{c(\phi)}{c(\theta)} + q\frac{s(\phi)}{c(\theta)}
\end{aligned}
$$

Force and torque apply to a Quadcopter

Total force acting on the quadcopter

$$m(\omega_b \wedge v_b + \dot{v_b}) = f_b, \quad \text{where} \quad f_b = mgR^T \hat{e}_z - f_t \hat{e}_3 + f_w$$

Total torque applied to the quadcopter

$$I\dot{\omega_b} + \omega_b \wedge (I\omega_b) = m_b, \quad \text{where} \quad m_b = \tau_B - g_a + \tau_w$$

$$\begin{aligned}
-mgs(\theta) + f_{wx} &= m(\dot{u} + qw + -rv) \\
mgc(\theta)s(\phi) + f_{wx} &= m(\dot{v} - pw + ru) \\
mgc(\theta)c(\phi) + f_{wz} - f_t &= m(\dot{w} + pv - qu) \\
\tau_x + \tau_{wx} &= \dot{p}I_x - qr(I_y - Iz) \\
\tau_y + \tau_{wy} &= \dot{q}I_y + pr(I_x - I_z) \\
\tau_z + \tau_{wz} &= \dot{r}I_z - pq(I_x - I_y)
\end{aligned}$$

Linearization around the equilibrium point

$$\bar{x} = [0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad \bar{x} \quad \bar{y} \quad \bar{z}]^T \quad \text{and} \quad \bar{u} = [mg \quad 0 \quad 0 \quad 0]^T$$

$$A = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -g & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ g & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \quad B = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & \frac{1}{I_x} & 0 & 0 \\ 0 & 0 & \frac{1}{I_y} & 0 \\ 0 & 0 & 0 & \frac{1}{I_z} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \frac{1}{m} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

State Space Matrix

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad \textit{and} \quad D = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

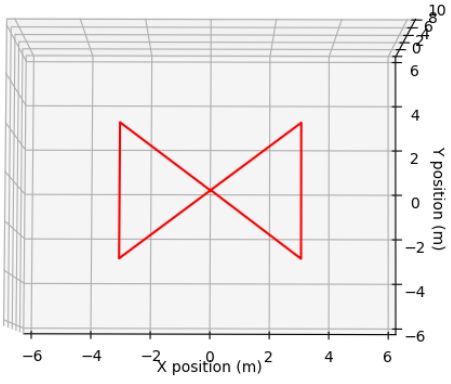
Model Predictive Control (MPC)

$$
\begin{aligned}
J &= J_e + J_u \\
J_e &= \sum_{k=1}^{N} \vec{e}^{\,T}(k) Q \vec{e}(k), \vec{e} = \overrightarrow{ref} - C\vec{x}(k) \\
J_u &= \sum_{k=0}^{N-1} \vec{U}^{\,T}(k) R \vec{U}(k) \\
\min_{U(k)} J &= \min_{U(k)} \frac{1}{2} \sum_{k=0}^{N-1} [\vec{e}^{\,T}(k) Q \vec{e}(k) + \vec{U}^{\,T}(k) R \vec{U}(k)]
\end{aligned}
$$

$$
\text{Constrain:} \begin{cases} \vec{x}(k+1) = A\vec{x}(k) + B\vec{U}(k) \\ x(0) = \vec{x}_0 \\ U_L \leq U \leq U_U \end{cases}
$$

Generate Waypoints

Set the six waypoints in space $[0,0,2],[-3,3,2],[-3,-3,2],[3,3,2],[3,-3,2],[0,0,2]$



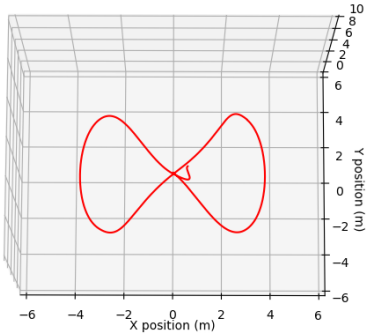
Cubic Spline Trajectory

Cubic Spline

$$
\begin{aligned}
x(u) &= a_x u^3 + b_x u^2 + c_x u + d_x \\
y(u) &= a_y u^3 + b_y u^2 + c_y u + d_y \\
z(u) &= a_z u^3 + b_z u^2 + c_z u + d_z
\end{aligned}
$$

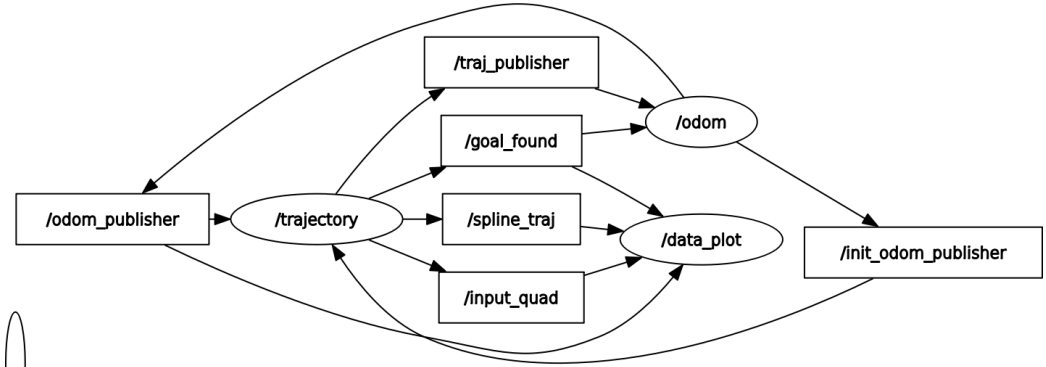
$$
\begin{aligned}
p(u) &= a u^3 + b u^2 + c u + d \\
&= \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \begin{bmatrix} a_x & a_y & a_z \\ b_x & b_y & b_z \\ c_x & c_y & c_z \\ d_x & d_y & d_z \end{bmatrix}
\end{aligned}
$$

with the same way points. We get:

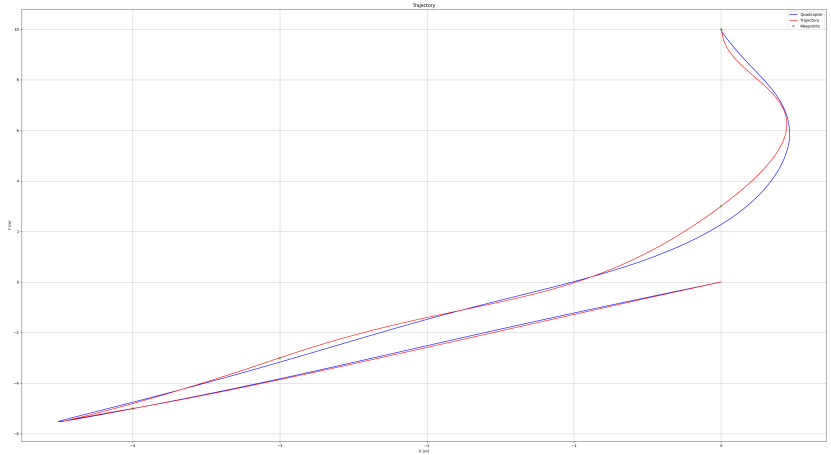


Rosgraph

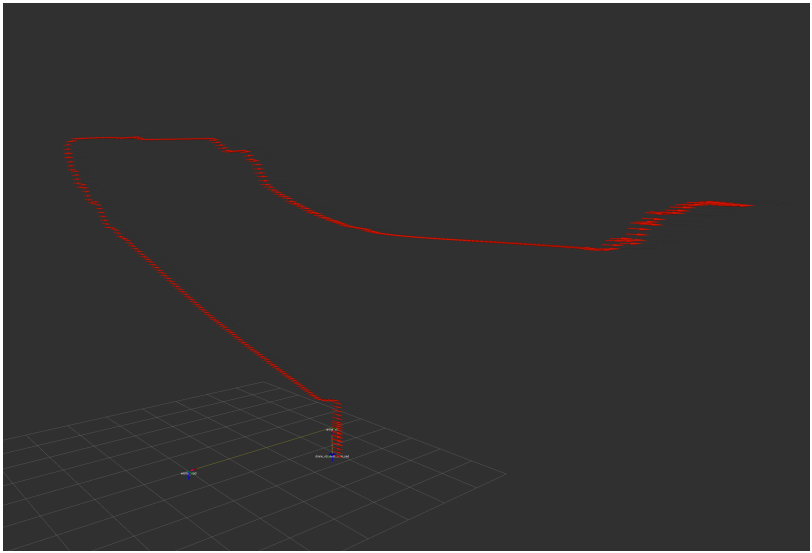
Rosgraph of the Trajectory Tracking of a Quadcopter



Tracking Graph



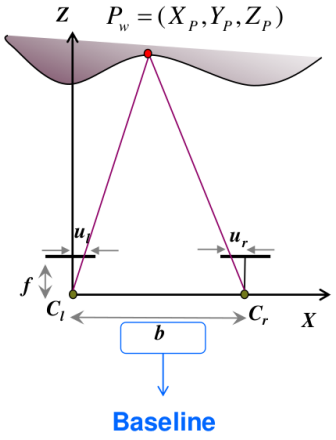
ROS Visualization (Rviz)



3. Depth Estimation

Stereo Vision

Stereo Vision is the process of obtaining the depth information from a pair of images coming from two cameras that look at the same scene from a different but known position.



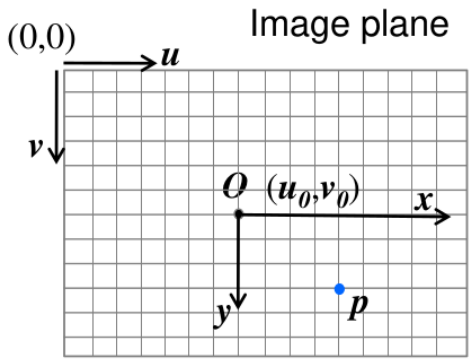
Depth Estimation

From Perspective Projection:

$$
\begin{aligned}
(u_l, v_l) &= (f_x \frac{X_P}{Z_P} + o_x, f_y \frac{Y_P}{Z_P} + o_y) \\
(u_r, v_r) &= (f_x \frac{X_P - b}{Z_P} + o_x, f_y \frac{Y_P}{Z_P} + o_y)
\end{aligned}
$$

Solving for X_P, Y_P and Z_P:

$$
\begin{aligned}
X_P &= \frac{b(u_l - o_x)}{u_l - u_r} \\
Y_P &= \frac{b f_x (u_l - o_y)}{f_y (u_l - u_r)} \\
Z_P &= \frac{b f_x}{u_l - u_r}
\end{aligned}
$$



4. Mapping an Environment

Realsense D435

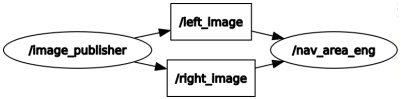
(a) Left Image

(b) Right Image

Depth and Navigation Area

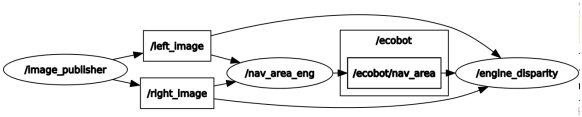
Depth Estimation:

Navigation Area:



Depth and Navigation Area

By combining Depth and Navigation Area:



Pointcloud Data

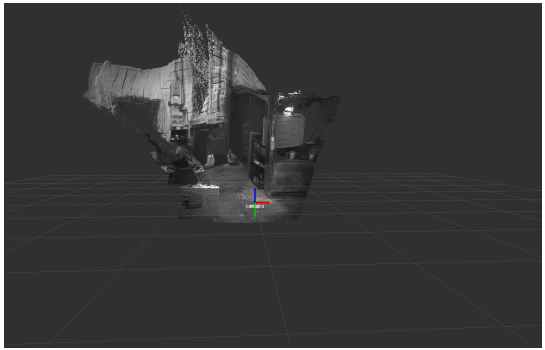
Convert Depth into pointcloud data:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = D \begin{bmatrix} \frac{1}{f_x} & 0 & 0 \\ 0 & \frac{1}{f_y} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$$

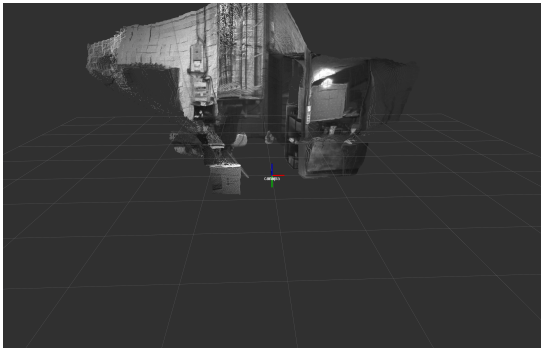
depth_image_proc: convert depth into pointcloud data

- Depth image
- Camera image
- Camera Information

Pointcloud Data



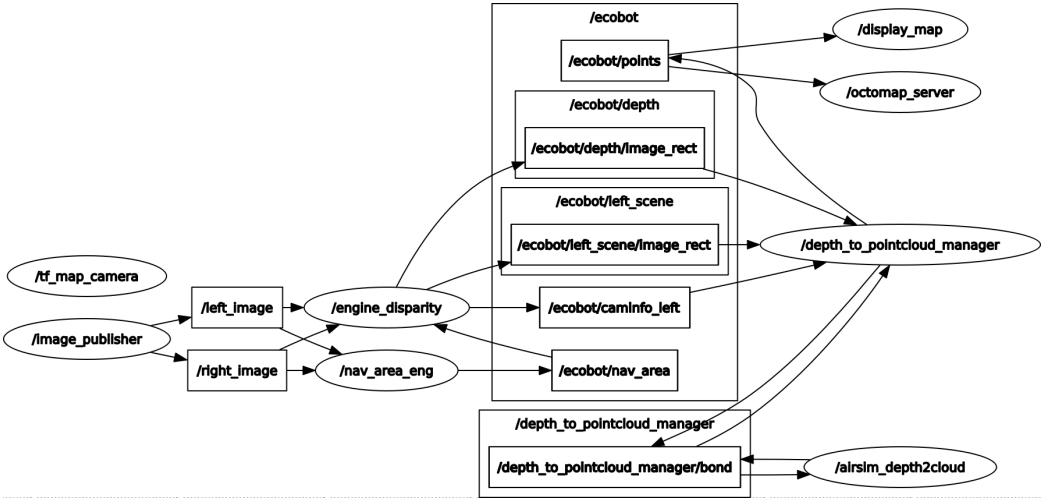
(a) Without Navigation Area



(b) With Navigation Area

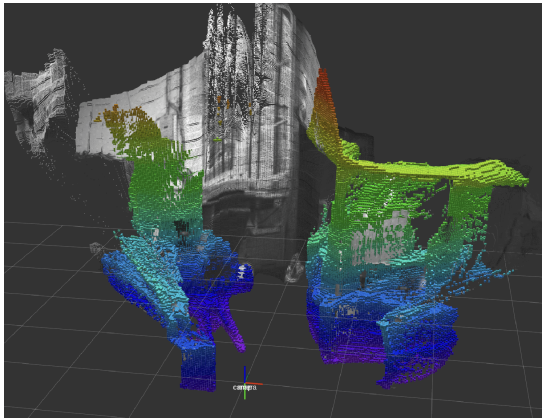
Rosgraph

Rosgraph of mapping environment using Realsense Camera



Mapping

3D and 2D Map



(a) 3D Map

(b) 2D Map

5. Navigation of a Mobile Robot

Gazebo Mapping

Mapping of an Environment in Gazebo:

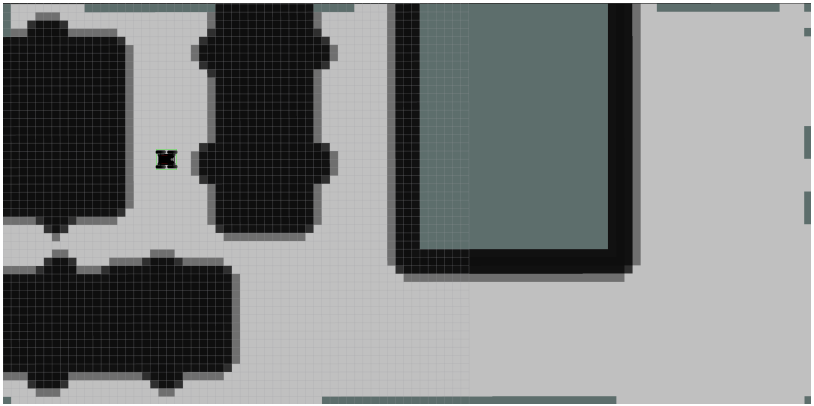
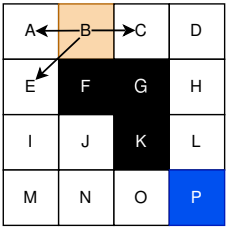


Figure: Occupancy Grid Map

Dijkstra Algorithm

Dijkstra is an algorithm which always keeps track of the shortest distance from the start node to each cell.

Navigation from node B to node P:



The iteration step of Dijkstra algorithm is defined by:

- Pick a current node:
- Neighbors of the current node:
- Update travel distance values.

After this iteration, the new cycle is repeated until reaching the target. Once the target is reached, the shorted route is extracted by every single node's parent node until reaching the start node.

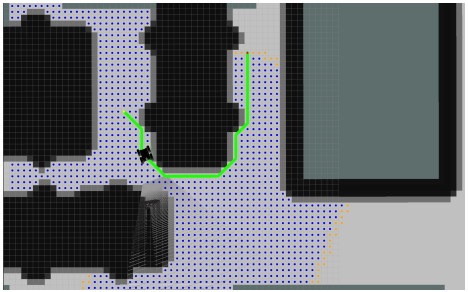
P	L	H	D	C	B

To have the path from the start node to the end node, the reverse is needed

B	C	D	H	L	P

Dijkstra with Dynamic Window Approach

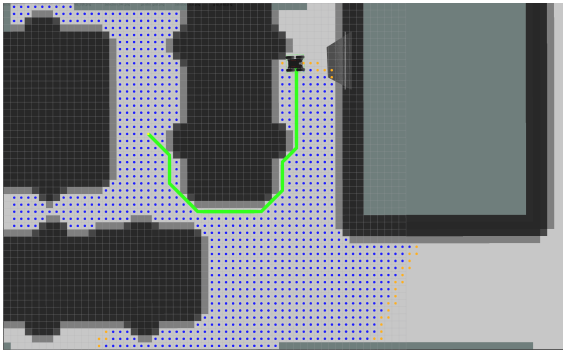
To simulate the robot in the gazebo environment using Dijkstra algorithm, the planer is needed to connect the path to the robot.



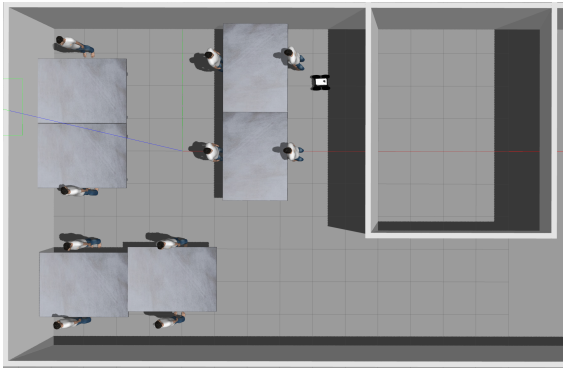
Dynamic Window Approach (DWA): Provide a controller that drives a mobile base in the plane. Using a map, the planner creates a kinematic trajectory for the robot to get from the start to the goal location.

Navigation of a Mobile Robot

Navigation Result:



(a) RVIZ



(b) Gazebo

6. Conclusion and Future Work

Conclusion

In this project:

- A mathematical model of Quadcopter dynamics is derived using Newton's and Euler's laws.
- A linearized version of the model is obtained.
- Using Model Predictive Control to track the trajectory.
- Using depth image of Airsim Environment to construct the 2D and 3D map of the environment.
- Using Realsense grayscale images to estimate depth and navigation area.
- Covert depth into pointcloud data and from pointcloud data into 2D and 3D map using Octomap server.
- Dijkstra algorithm with Dynamic Window Approach to navigate the robot in the environment.

Future Work

- Applying Quadcopter model and MPC into the real-life Quadcopter.
- Mapping the environment with Rover Robotics using Realsense Camera
- Navigation and Mapping the environment of a Mobile robot at the same time.

THANK YOU

Linearization

$$
\begin{aligned}
\dot{\phi} &= p + r\theta + q\phi\theta \\
\dot{\theta} &= q - r\phi \\
\dot{\psi} &= r + q\phi \\
\dot{p} &= \frac{I_y - I_z}{I_x} rq + \frac{\tau_x - \tau_{wx}}{I_x} \\
\dot{q} &= \frac{I_z - I_x}{I_y} pr + \frac{\tau_y + \tau_{wy}}{I_y} \\
\dot{r} &= \frac{I_x - I_y}{I_z} pq + \frac{\tau_z + \tau_{wz}}{I_z}
\end{aligned}
$$

$$
\begin{aligned}
\dot{u} &= rv - qw - g\theta + \frac{f_{wx}}{m} \\
\dot{v} &= pw - ru + g\phi + \frac{f_{wy}}{m} \\
\dot{w} &= qu - pv + g + \frac{f_{wz} - f_t}{m} \\
\dot{x} &= w(\phi\psi + \theta) - v(\psi - \phi\theta) + u \\
\dot{y} &= v(1 + \phi\theta\psi) - w(\phi - \psi\theta) + u\psi \\
\dot{z} &= w + v\phi - u\theta
\end{aligned}
$$

MPC cost and constrain

```python
def run_mpc(self, rx,x,u,x_init):
    cost = 0.
    umin =np.array([7.7 , 7.7 , 7.7 , 7.7])-self.u0
    umax =np.array([13. , 13. , 13. , 13.])-self.u0
    INF = np.inf
    xmin = np.array([-0.2, -0.2, -2*np.pi, -.25, -.25, -.25,  -INF,  -INF,  -INF, -INF, -INF, -INF])
    xmax = np.array([0.2,  0.2,   2*np.pi,  .25, .25,  .25,   INF,   INF,   INF,  INF,  INF, INF])

    constr = [x[:, 0] == x_init]
    for t in range(self.N):
        cost += cp.quad_form(rx - x[:, t], self.Q) + cp.quad_form(u[:, t], self.R)  # Linear Quadratic cost
        constr += [xmin <= x[:, t], x[:, t] <= xmax]  # State constraints
        constr += [umin <= u[:, t], u[:, t] <= umax]
        constr += [x[:, t + 1] == self.A_zoh * x[:, t] + self.B_zoh * u[:, t]]

    cost += cp.quad_form(x[:, self.N] - rx, self.Q)  # End of trajectory error cost
    problem = cp.Problem(cp.Minimize(cost), constr)
    return problem
```

Update State

From newton's law:

$$m\dot{v} = Rf_B = mg\hat{e_z} - f_t R\hat{e_3}$$

$$\begin{cases} \ddot{x} = -\dfrac{f_t}{m}[s(\phi)s(\psi) + c(\phi)c(\psi)s(\theta)] \\ \ddot{y} = -\dfrac{f_t}{m}[c(\phi)s(\psi)s(\theta) - c(\psi)s(\phi)] \\ \ddot{z} = g - \dfrac{f_t}{m}[c(\phi)c(\theta)] \end{cases}$$

Now a simplified is made by setting $[\dot{\phi} \quad \dot{\theta} \quad \dot{\psi}] = [p \quad q \quad r]$ is true for small angle.

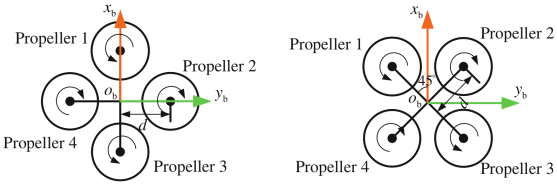
$$\begin{cases} \ddot{\phi} = \dfrac{I_y - I_z}{I_x}\dot{\theta}\dot{\psi} + \dfrac{\tau_x - \tau_{wx}}{I_x} \\ \ddot{\theta} = \dfrac{I_z - I_x}{I_y}\dot{\phi}\dot{\psi} + \dfrac{\tau_y + \tau_{wy}}{I_y} \\ \ddot{\psi} = \dfrac{I_x - I_y}{I_z}\dot{\phi}\dot{\theta} + \dfrac{\tau_z + \tau_{wz}}{I_z} \end{cases}$$

Linearize the newton's law

$$\begin{cases} \ddot{\phi} = \dfrac{\tau_x}{I_x} \\ \ddot{\theta} = \dfrac{\tau_y}{I_y} \\ \ddot{\psi} = \dfrac{\tau_z}{I_z} \\ \ddot{x} = -g\theta \\ \ddot{y} = g\phi \\ \ddot{z} = -(g - \dfrac{f_t}{m}) \end{cases}$$

Trajectory Tracking of a Quadcopter

Thrust and Moments Model



The input that can be applied to the system in order to control the behavior of the quadcopter

$$
\begin{aligned}
f_t &= b(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2) \\
\tau_x &= bl(-\Omega_1^2 + \Omega_2^2 + \Omega_3^2 - \Omega_4^2) \\
\tau_y &= bl(\Omega_1^2 - \Omega_2^2 + \Omega_3^2 - \Omega_4^2) \\
\tau_z &= d(\Omega_1^2 + \Omega_2^2 - \Omega_3^2 - \Omega_4^2)
\end{aligned}
$$

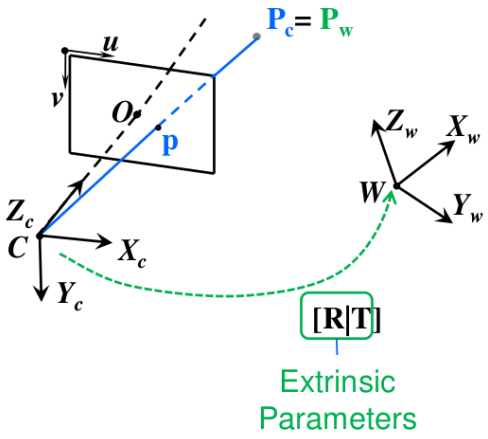
Weight matrix Q and R

Q and R has a more immediate effect over the behavior of the quadcopter. The value of Q will remain constant while the value of R will be be used to tune the MPC.

- The first value affects the throttle command.
- The second value affects the roll.
- The third value affects the pitch.
- The fourth value affects the yaw.

The lower values would mean a faster system because the controller will use higher command, but it will result in bigger overshoot or even an unstable close loop system.

World to Camera Coordinate

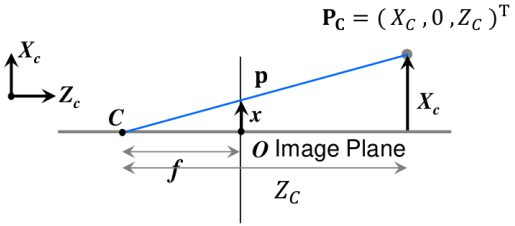


$$\begin{bmatrix} X_C \\ Y_C \\ Z_C \\ 1 \end{bmatrix} = \begin{bmatrix} R_{3\times3} & t_{3\times1} \\ 0_{1\times3} & 1_{1\times1} \end{bmatrix}_{4\times4} \begin{bmatrix} X_W \\ Y_W \\ Z_W \\ 1 \end{bmatrix}$$

$$= [R|T] \begin{bmatrix} X_W \\ Y_W \\ Z_W \\ 1 \end{bmatrix}$$

- where $[R|T]$ is the camera extrinsic matrix with the rotation R and transition T operation.

Camera Coordinate to Image Coordinate



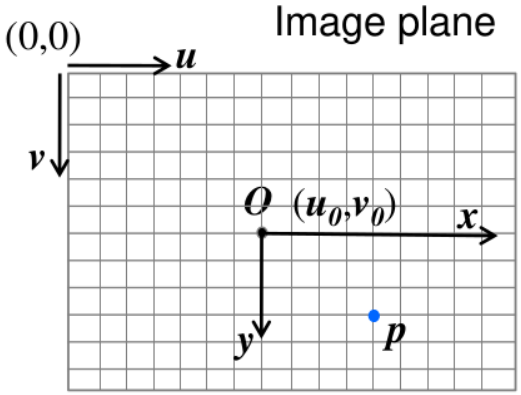
From similar triangle:

$$\frac{x}{f} = \frac{X_C}{Z_C} => x = \frac{fX_C}{Z_C}$$
$$\frac{y}{f} = \frac{Y_C}{Z_C} => y = \frac{fY_C}{Z_C}$$

Hence the following transformation matrix from the camera coordinate system to image coordinate system is:

$$\begin{bmatrix} x \\ y \\ Z_C \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X_C \\ Y_C \\ Z_C \\ 1 \end{bmatrix}$$

Image coordinate to Pixel



$$
\begin{aligned}
u &= u_0 + kx = u_0 + k\frac{fX_C}{Z_C} \\
v &= v_0 + ky = v_0 + k\frac{fY_C}{Z_C}
\end{aligned}
$$

Expressed in matrix form and homogeneous coordinate

$$
\begin{bmatrix} \lambda u \\ \lambda v \\ \lambda \end{bmatrix} = \begin{bmatrix} kf & 0 & u_0 \\ 0 & kf & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_C \\ Y_C \\ Z_C \end{bmatrix} = K \begin{bmatrix} X_C \\ Y_C \\ Z_C \end{bmatrix}
$$

Trajectory Tracking of a Quadcopter

Cubic Polynomial

$$\begin{aligned} p(u) &= au^3 + bu^2 + cu + d \\ &= [u^3 \quad u^2 \quad u \quad 1][a \quad b \quad c \quad d] \end{aligned}$$

Three cubic polynomial, one for each coordinate

$$\begin{aligned} x(u) &= a_x u^3 + b_x u^2 + c_x u + d_x \\ y(u) &= a_y u^3 + b_y u^2 + c_y u + d_y \\ z(u) &= a_z u^3 + b_z u^2 + c_z u + d_z \end{aligned}$$

$$\begin{aligned} p(u) &= au^3 + bu^2 + cu + d \\ &= [u^3 \quad u^2 \quad u \quad 1] \begin{bmatrix} a_x & a_y & a_z \\ b_x & b_y & b_z \\ c_x & c_y & c_z \\ d_x & d_y & d_z \end{bmatrix} \end{aligned}$$