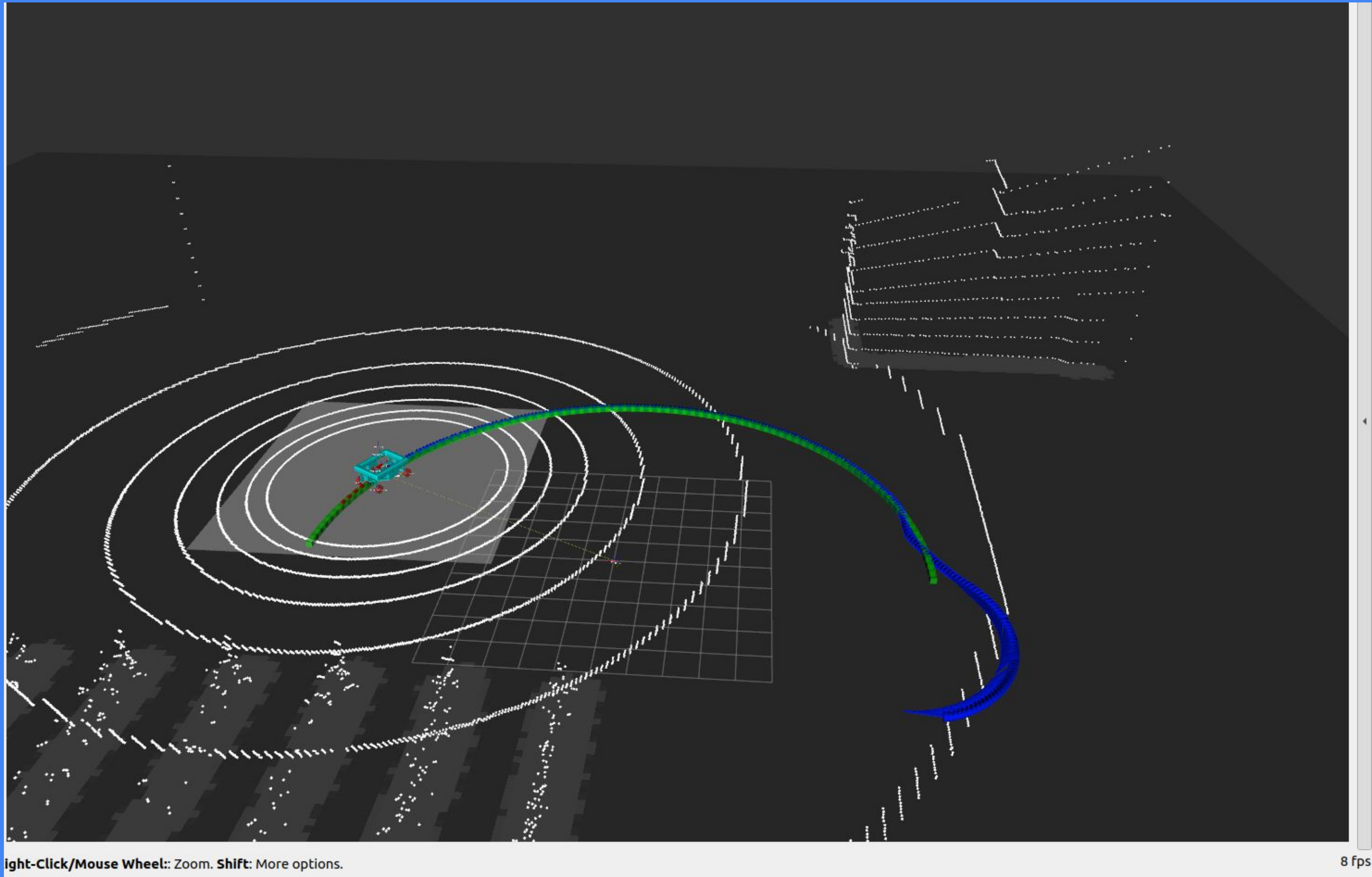
Trajectory Control of Ackermann Steering Robot With Nonlinear MPC



Deriving the kinematics of simplified bicycle model

$$m = \text{mass of the vehicle}$$
$$l_f, l_r = \text{Distance from CoG to Front/Rear Axels}$$
$$I_z = \text{Moment of Inertia of vehicle about z axis}$$
$$x = \text{Longitudinal position [global frame]}$$
$$y = \text{Lateral position [global frame]}$$
$$v_x = \text{Longitudinal velocity [body frame]}$$
$$v_y = \text{Lateral velocity [body frame]}$$
$$\dot{\psi} = r = \text{Yaw rate}$$
$$\delta = \text{Front steering angle}$$
$$F_{xR} = \text{Input rear force}$$
$$F_{xF}, F_{yF} = \text{Longitudinal/Lateral force on front tire}$$

Robot's coordinate frames

Deriving the kinematics of simplified bicycle model

State dynamics then become;

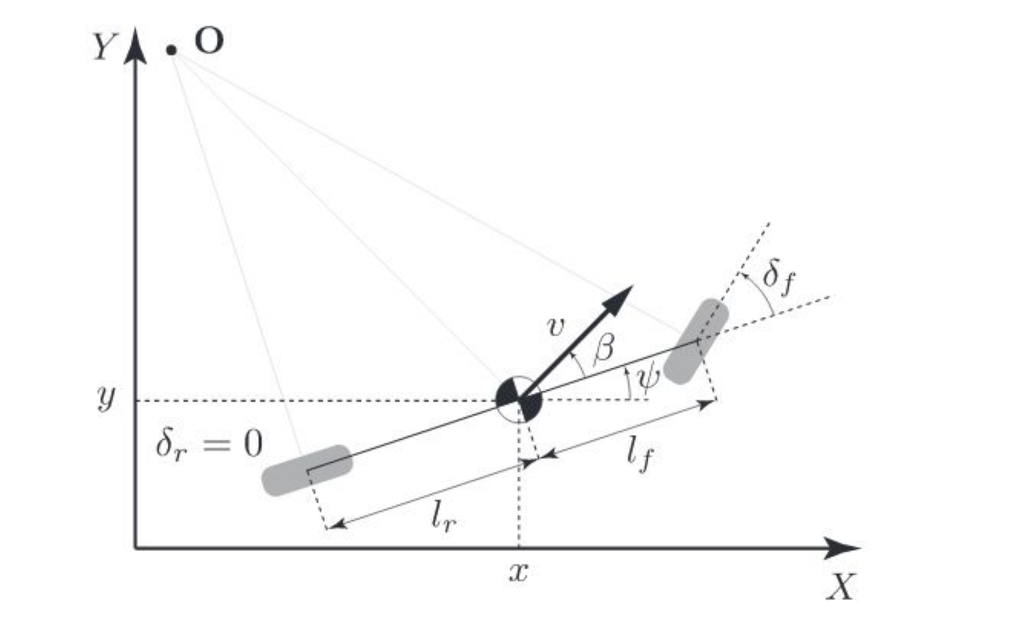
$$\dot{x} = v\ \cos(\psi + \beta)$$

$$\dot{y} = v\ \sin(\psi + \beta)$$

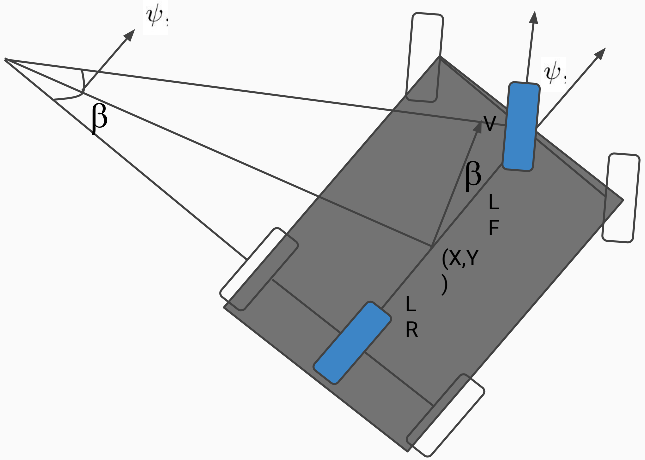
$$\dot{\psi} = \frac{v\ \cos(\beta)}{l_f + l_r}\ \tan(\delta_f) = \frac{v}{l_r}\sin(\beta)$$

$$\dot{v} = a$$

$$\beta = \tan^{-1}\left(\frac{l_r}{l_f + l_r}\ \tan(\delta_f)\right)$$



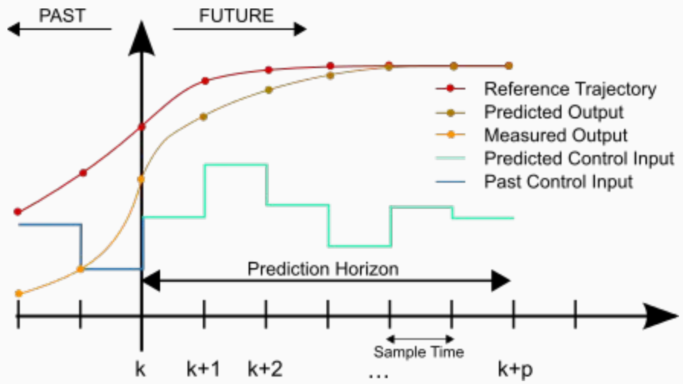
Discretized state dynamics in C++




```
auto beta = casadi::MX::atan(L_R / (L_F + L_R) * casadi::MX::tan(df_dv_(i)));

opti_->subject_to(
  x_dv_(i + 1) == x_dv_(i) + DT * (v_dv_(i) * casadi::MX::cos(psi_dv_(i) + beta)));

opti_->subject_to(
  y_dv_(i + 1) == y_dv_(i) + DT * (v_dv_(i) * casadi::MX::sin(psi_dv_(i) + beta)));

opti_->subject_to(
  psi_dv_(i + 1) == psi_dv_(i) + DT * (v_dv_(i) / L_R * casadi::MX::sin(beta)));

opti_->subject_to(
  v_dv_(i + 1) == v_dv_(i) + DT * acc_dv_(i));
```

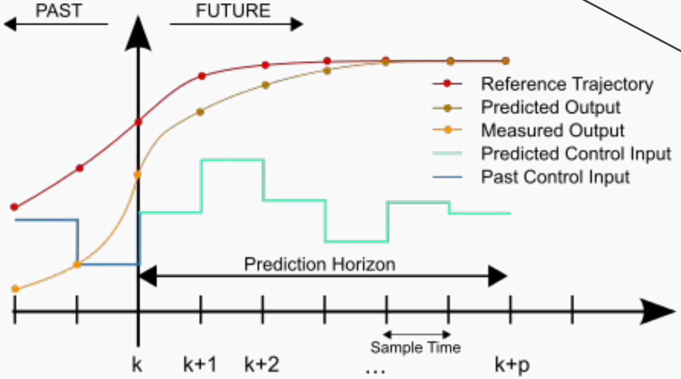
Model Predictive Control(MPC)

- Advanced method to control a process while satisfying some constraints
- Rely on dynamic models of the process
- MPC is based on iterative, finite-horizon optimization of a plant model



Model Predictive Control(MPC)

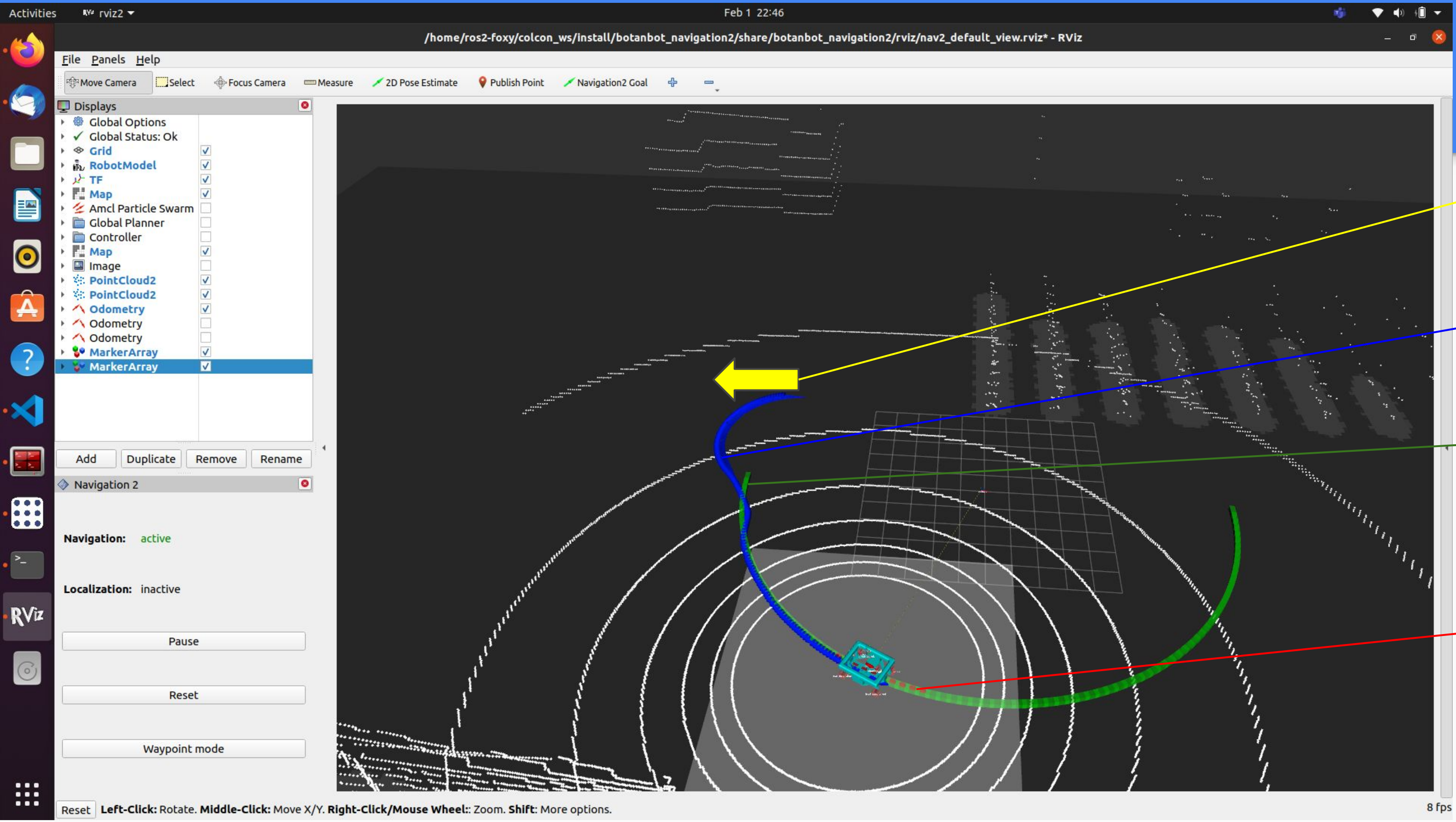
- Given Quadratic cost function J, minimize J subject to constraints



$$\begin{aligned}
\min_{\mathbf{U}} \quad & \sum_{i=0}^{H_p} (\mathbf{z}_i - \mathbf{z}_{ref,i})^T Q (\mathbf{z}_i - \mathbf{z}_{ref,i}) + && \text{(4a)} \\
& \sum_{i=0}^{H_p-1} \left[(\mathbf{u}_i - \mathbf{u}_{i-1})^T \bar{R} (\mathbf{u}_i - \mathbf{u}_{i-1}) + \mathbf{u}_i^T R \mathbf{u}_i \right] && \\
\text{s.t.} \quad & \mathbf{z}_0 = \mathbf{z}(t), \mathbf{u}_{-1} = \mathbf{u}(t - t_s) && \text{(4b)} \\
& \mathbf{z}_{i+1} = f(\mathbf{z}_i, \mathbf{u}_i), \; i = 0, \ldots, H_p - 1 && \text{(4c)} \\
& \mathbf{u}_{min,i} \leq \mathbf{u}_i \leq \mathbf{u}_{max,i}, \; \forall i && \text{(4d)} \\
& \dot{\mathbf{u}}_{min,i} \leq \frac{\mathbf{u}_i - \mathbf{u}_{i-1}}{\mathrm{t}_d} \leq \dot{\mathbf{u}}_{max,i}, \; \forall i \backslash 0 && \text{(4e)} \\
& \dot{\mathbf{u}}_{min,i} \leq \frac{\mathbf{u}_i - \mathbf{u}_{i-1}}{\mathrm{t}_s} \leq \dot{\mathbf{u}}_{max,i}, \; i = 0 && \text{(4f)}
\end{aligned}$$

Z_(ref,i) : Reference Traj
Z_(i) : Predicted Traj
U_(u-i) : Previous Control input
U_(u) : Predicted Control input

Results



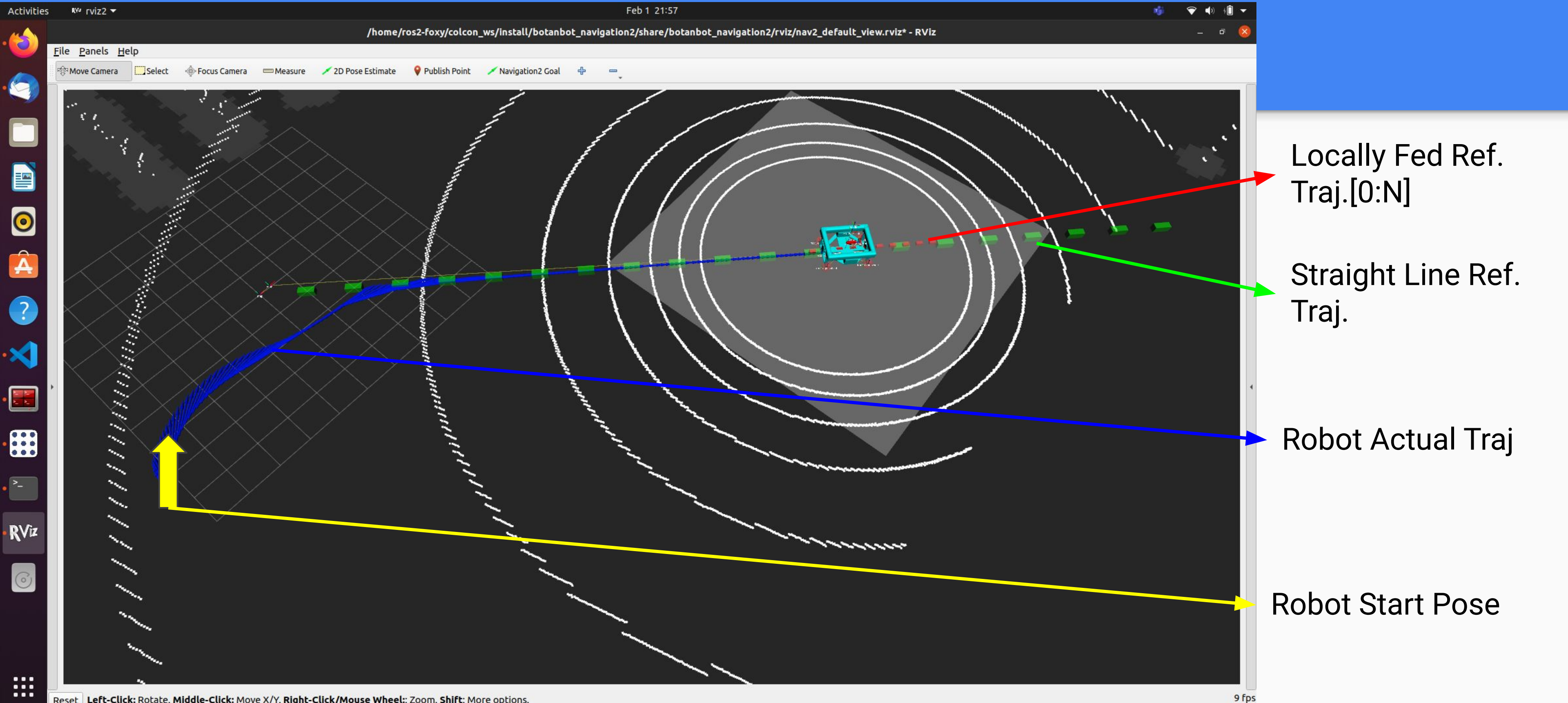
Robot Start Pose

Robot Actual Traj

Half Circle Ref. Traj.

Locally Fed Ref. Traj.[0:N]

Results


Activities
rviz2
Feb 1 21:57
/home/ros2-foxy/colcon_ws/install/botanbot_navigation2/share/botanbot_navigation2/rviz/nav2_default_view.rviz* - RViz
File Panels Help
Move Camera
Select
Focus Camera
Measure
2D Pose Estimate
Publish Point
Navigation2 Goal
Reset
Left-Click: Rotate. Middle-Click: Move X/Y. Right-Click/Mouse Wheel:: Zoom. Shift: More options.
9 fps
Locally Fed Ref. Traj.[0:N]
Straight Line Ref. Traj.
Robot Actual Traj
Robot Start Pose

Video

```cpp
/**
 * @brief all parameters usd by MPC class,
 * user needs to create and reconfigure this
 *
 */
struct Parameters
{
  // timesteps in MPC Horizon
  int N;
  // discretization time between timesteps(s)
  double DT;
  // distance from CoG to front axle(m)
  double L_F;
  // distance from CoG to rear axle(m)
  double L_R;
  // min / max velocity constraint(m / s)
  double V_MIN;
  double V_MAX;
  // min / max acceleration constraint(m / s ^ 2)
  double A_MIN;
  double A_MAX;
  // min / max front steer angle constraint(rad)
  double DF_MIN;
  double DF_MAX;
  // min / max jerk constraint(m / s ^ 3)
  double A_DOT_MIN;
  double A_DOT_MAX;
  // min / max front steer angle rate constraint(rad / s)
  double DF_DOT_MIN;
  double DF_DOT_MAX;
  // weights on x, y, psi, and v.
  std::vector<double> Q;
  // weights on jerk and slew rate(steering angle derivative)
  std::vector<double> R;
  // enable/disable debug messages
  bool debug_mode;
  // set this true only if user figured the configration
  bool params_configured;
```

Next

```
#goal definition
nav_msgs/Path path
string controller_id
---
#result definition
std_msgs/Empty result
---
#feedback
float32 distance_to_goal
float32 speed
```

- Add Collisions as constraints
- Expose MPC controller as an ***FollowPath.action***
- Tune MPC Controllers **Parameters**

References;

Kinematic and dynamic vehicle models for autonomous driving control design
Engineering, Computer Science
2015 IEEE Intelligent Vehicles Symposium (IV)