Point Stabilization of Non-Holonomic Robot using MPC

Nisarga Nilavadi
M.Sc. Embedded Systems Engineering
MatrNr. 5169068

Premraj Thakur
M.Sc. Embedded Systems Engineering
MatrNr. 5369559

Robinson Pompeu
M.Sc. Mathematics
MatrNr. 5455601

***Abstract*—This paper discusses an optical control problem of driving a non-holonomic robot from an initial point to the target point (point stabilization), under different scenarios such as presence of obstacles, bounds on speed, etc. using model predictive control. The Optical Control problem is first reduced to corresponding NLP problem using two different methods, Single shooting and multiple shooting followed by implementation of both methods with CasADi. Finally, results are analysed and compared.**

Contents

***Index Terms*—Mobile Robot, Velocity based kinematics, Point Stabilization, Model predictive control, Optimal Control, Non Linear Programming, Single-Shooting, Multiple-Shooting, Matlab, CasADi, IPOPT.**

I. Introduction

Autonomous navigation enables an agent to plan its path and execute plan without human intervention. In the recent technological advancements, this research area is potentially applied in wide range of applications like driver-less cars, humanoid robots, military systems, also to explore the outer space part of our universe e.g,. Mars rover autonomous vehicle. In robotic application, planning task reduces down to solving optimal control problem on the system dynamics to obtain control actions that abide by the constraints rules minimizing the objective function to achieve a task. One such example is point stabilization of a mobile robot.

II. Mathematical model

A. System Dynamics

The Configuration of the robot on a two dimentional planar surface is defined by state vector X, which is a three dimentional vector consisting orthonormal coordinates (x, y) and angle θ. while x and y define the position of robot, θ indicates the angle at which the robot is tilted. **Therefore, system state is represented by** (1)

$$\boldsymbol{X} = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} \qquad \begin{matrix} x, y \text{ in meters} \\ \theta \text{ in radians} \end{matrix} \tag{1}$$

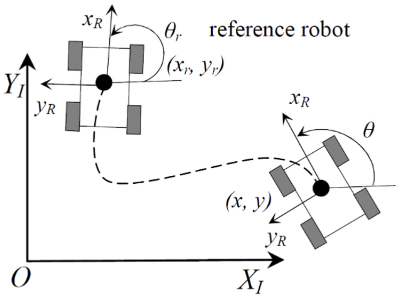

Fig. 1. Description of the state in terms of coordinates x, y and θ.

We denote by ϕ_r and ϕ_ℓ the velocities (in meters per second) of the right and left wheels, respectively. Note that in this case translational velocity of the robot is given by (2).

$$v = \frac{r}{2}\left(\dot{\phi_r} + \dot{\phi_\ell}\right) \tag{2}$$

and the angular velocity of the robot is given by (3).

$$\omega = \frac{r}{2D}\left(\dot{\phi_r} - \dot{\phi_\ell}\right) \tag{3}$$

where,
r is the radius of each wheel in meters.

D is the distance between the left and right wheels in meters.

The controls are therefore given by (4)

$$\boldsymbol{U} = \begin{bmatrix} v \\ \omega \end{bmatrix} \qquad \begin{matrix} v \text{ in m/s.} \\ \theta \text{ in radians/s.} \end{matrix} \tag{4}$$

Kinematics is the description of the effect of control actions on the configuration of a robot. **The dynamics of the system** can be described by the first order ODE system and is given by (5).

$$\begin{gathered} \dot{\boldsymbol{x}}(t) = f\big(\boldsymbol{x}(t), \boldsymbol{u}(t)\big) \\ \begin{bmatrix} \dot{x}(t) \\ \dot{y}(t) \\ \dot{\theta}(t) \end{bmatrix} = \begin{bmatrix} v\cos\theta(t) \\ v\sin\theta(t) \\ \omega(t) \end{bmatrix}. \end{gathered} \tag{5}$$

B. Discretization methods

The system time evolution given in (5) can be discretized with Euler method. For the discrete evolution, the iteration step is given by (6).

$$\begin{gathered} \boldsymbol{X}_{k+1} = f\left(\boldsymbol{X}_k, \boldsymbol{U}_k\right) \\ \begin{bmatrix} X_{k+1} \\ Y_{k+1} \\ \theta_{k+1} \end{bmatrix} = \begin{bmatrix} X_k \\ Y_k \\ \theta_k \end{bmatrix} + \Delta t \begin{bmatrix} v_k\cos\theta_k \\ v_k\sin\theta_k \\ \omega_k \end{bmatrix}. \end{gathered} \tag{6}$$

When the Horizon Length (N) increases, it is observed that discretization accuracy of Euler deteriorates drastically. Hence Runge kutta method is used for longer N. For Runge-Kutta method, iteration step is given by (7).

$$\begin{aligned} k_1 &= X_k + \Delta t \begin{bmatrix} v_k\cos\theta_k \\ v_k\sin\theta_k \\ \omega_k \end{bmatrix} \\ k_2 &= X_k + \frac{\Delta t}{2} k_1 + \begin{bmatrix} v_k\cos\theta_k \\ v_k\sin\theta_k \\ \omega_k \end{bmatrix} \\ k_3 &= X_k + \frac{\Delta t}{2} k_2 + \begin{bmatrix} v_k\cos\theta_k \\ v_k\sin\theta_k \\ \omega_k \end{bmatrix} \\ k_4 &= X_k + \Delta t k_3 + \begin{bmatrix} v_k\cos\theta_k \\ v_k\sin\theta_k \\ \omega_k \end{bmatrix} \\ X_{K+1} &= X_k + \frac{\Delta t}{6}(k_1 + 2k_2 + 2k_3 + k_4) \end{aligned} \tag{7}$$

where $k = 0, 1, \ldots, N-1$.

III. Optimal Control Problem

A. Model predictive control

Model Predictive control entails solving Optimal Control Problems in an iterative manner. At each stage of the iteration, we solve an OCP problem to obtain an optimal sequence of controls which intend to take the dynamic system to the desired state in certain number of time-steps, termed as Prediction horizon (N). However, only the first control is applied and the remaining sequence of controls, shifted by one horizon, is taken as an initial point for the next iteration.

B. OCP formulation

As universal in the realm of Numerical Optimal control we convert the OCP into an equivalent NLP problem with an objective function, optimization variables and constraints. The general OCP formulation for our case is given by (8).

$$\begin{aligned} \underset{X,U \text{ admissible}}{\text{minimize}} \ & \sum_{k=0}^{N-1} \ell\big(\boldsymbol{X}_k, \boldsymbol{U}_k\big) \\ \text{subject to } & \boldsymbol{X}_{k+1} = f\big(\boldsymbol{X}_k, \boldsymbol{U}_k\big), k \in \{1, \ldots, N\} \\ & v_{min} \le \boldsymbol{v}_k \le v_{max} \\ & \theta_{min} \le \boldsymbol{\omega}_k \le \theta_{max} \\ & x_{min} \le \boldsymbol{x}_k \le x_{max} \\ & y_{min} \le \boldsymbol{y}_k \le y_{max} \\ & \theta_{min} \le \theta_k \le \theta_{max}. \end{aligned} \tag{8}$$

where,

$$\begin{aligned} \ell\big(\boldsymbol{X}_k, \boldsymbol{U}_k\big) &= \|X_k - X_{ref}\|^2 + \|U_k - U_{ref}\|^2 \\ &= \left\| \begin{bmatrix} x_k \\ y_k \\ \theta_k \end{bmatrix} - \begin{bmatrix} x_{ref} \\ y_{ref} \\ \theta_{ref} \end{bmatrix} \right\|^2 + \left\| \begin{bmatrix} v_k \\ w_k \end{bmatrix} - \begin{bmatrix} v_{ref} \\ w_{ref} \end{bmatrix} \right\|^2 \end{aligned}$$

N = 30 is the prediction Horizon.
$X_0 = [0, 0, 0]^T$ is the initial state.
$X_{ref} = [1.8, 1.2, \pi]^T$ is the desired target state.
$[v_{ref}, w_{ref}] = [0, 0]^T$ is the reference control.

We reduce this OCP problem to corresponding NLP problem with equality and inequality constraints. Casadi's IpOpt solver is used to solve this NLP iteratively until loop exit condition is satisfied.

IV. NLP formulation

We have used the two most common strategies to reduce OCP to NLP and explored Single shooting (sequential) and Multiple shooting (simultaneous) methods.

A. Single shooting

In single shooting, we eliminate all the state variables by expressing them as a function of control variables and solve the optimization problem with only the control variables as the optimization variables. As such a state X_k is expressed by the recursive expression: $f(f(\ldots f(f(X_0, U_0), U_1), \ldots U_{k-1})$. Hence, the state variables are now functions of control inputs. Since, we have used the system dynamics to express the states in terms of the controls, now the NLP problem can be written as in (9).

$$\begin{aligned} \underset{U \text{ admissible}}{\text{minimize}} \ & \sum_{k=0}^{N-1} \ell\big(\boldsymbol{X}_k, \boldsymbol{U}_k\big) \\ \text{subject to } & -1.6 \text{ m/s} \le \boldsymbol{v}_k \le 1.6 \text{ m/s} \\ & -\pi/4 \text{ radians/s} \le \boldsymbol{\omega}_k \le \pi/4\text{radians/s} \\ & -2 \text{ m } \le (x_k)_u \le 2 \text{ m} \\ & -2 \text{ m } \le (y_k)_u \le 2 \text{ m} \\ & -\infty \text{ radians } \le (\theta_k)_u \le \infty \text{ radians.} \end{aligned} \tag{9}$$

where

$$[(x_u)_k, (y_u)_k, (\theta_u)_k]^T = f(X_0, U_k)$$

Owing to the nested function calls, single shooting method yields high error in cases where the system dynamics is highly non-linear due to non-linear error propagation over N horizon length.

B. Multiple shooting

In Multiple shooting, we divide the prediction horizon into subsections and the system dynamics is integrated in parallel in each of the subsections. In terms of NLP, both the intermediate states and the control variables are the optimization variables. Hence the NLP formulation is given by (10).

$$\begin{aligned}
&\underset{X,U \text{ admissible}}{\text{minimize}} \sum_{k=0}^{N-1} \ell\left(\boldsymbol{X}_k, \boldsymbol{U}_k\right) \\
&\text{subject to } \boldsymbol{X}_{K+1} = f\left(\boldsymbol{X}_k, \boldsymbol{U}_k\right), k \in \{1, \dots, N\} \\
&\qquad - 1.6 \text{ m/s} \leq \boldsymbol{v}_k \leq 1.6 \text{ m/s} \\
&\qquad - \pi/4 \text{ radians/s } \leq \boldsymbol{\omega_k} \leq \pi/4\text{radians/s} \\
&\qquad - 2 \text{ m } \leq \boldsymbol{x}_k \leq 2 \text{ m} \\
&\qquad - 2 \text{ m } \leq \boldsymbol{y}_k \leq 2 \text{ m} \\
&\qquad - \infty \text{ radians } \leq \theta_k \leq \infty \text{ radians.} \qquad (10)
\end{aligned}$$

As we now take both the state variables and control variables as the variables in the optimization problem, the dimensionality of the optimization problem is lifted and this results in faster convergence, however the computation time increases. In addition to that, as the ODE's are solved in parallel in each time horizon, there is an increase in performance.

C. Obstacles Constraint Formulation

A circular obstacle is introduced in the global coordinate frame (environment) along the path from initial point to final point. $h(x)$ defined in (11) is the formulation of an extra constraint added to NLP which is an euclidean distance between two circular objects, robot and obstacle.

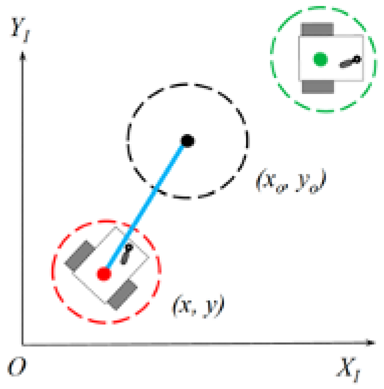

Fig. 2. Eucledean distance between robot and obstacle.

$$\begin{aligned}
h(x) = &\sqrt{(x - x_0)^2 + (y - y_0)^2} \geq r_r + r_0 \\
&\sqrt{(x - x_0)^2 + (y - y_0)^2} - (r_r + r_0) \geq 0 \\
-&\sqrt{(x - x_0)^2 + (y - y_0)^2} + (r_r + r_0) \leq 0 \qquad (11)
\end{aligned}$$

where x and y are components of the state of robot which keeps changing in each iteration of MPC, this point is referred to as centre of mass of robot and r_r is radius of circular boundary drawn around robot dimensionality. x_0 and y_0 define centre of circular obstacle. r_0 define radius of circular obstacle. While implementing, stack of circular obstacles are used to create maze environment.

V. Demonstration and results

A. Parameters used in implementation

Before we dive deep into the result analysis, let us consider following parameters to understand the response of the model when changed over these parameters.
General algorithm parameters are as follows:
Prediction Horizon Length: N (20 to 30)
Sampling time: 0.2 s
Robot diameter = obstacle diameter = 0.2 m
Loop parameters are as follows:
Maximum iterations: 100
MPC iterations runs: mpciter
Error: $\|x_0 - x_s\|^2$

B. Comparing Single and multiple shooting Simulations

When robot is simulated to drive to a final point without obstacles on the way, the predicted control actions of single and Multiple shooting looks almost similar. Both Single shooting and Multiple shooting takes same trajectory as shown in Fig. 3.

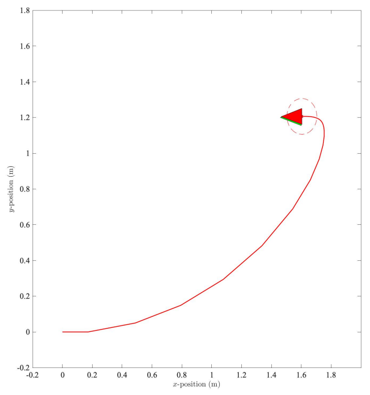

Fig. 3. Trajectory for Single shooting and multiple Shooting with $-1.6 < v < 1.6$ and $\theta_{\text{ref}} = \pi$.

Fig. 4 and Fig. 5 evidently shows that the control actions generated by both methods mostly look similar. However, the convergence rate is slightly faster in multiple shooting than single shooting. While single shooting took 37 iterations of control actions generations, multiple shooting took 32 iterations which can be keenly observed in Fig. 4 and Fig. 5.

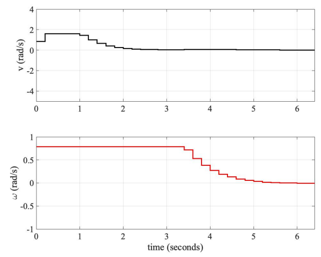

Fig. 4. Single shooting control actions.

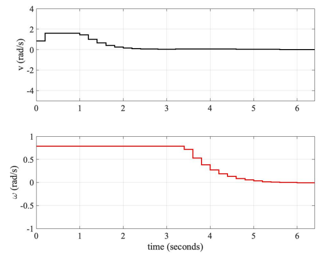

Fig. 5. Multiple shooting control actions.

C. *Multiple shooting simulations over constraints*

Fig. 6 shows the path taken by robot to reach final point from control actions when subjected to smaller velocity range i.e -0.6m/s $< v <$ 0.6m/s. It is shorter and has consistent uniform velocity from the initial point to reach final point without turns. Convergence is in 30 iterations, error encountered is 0.008.

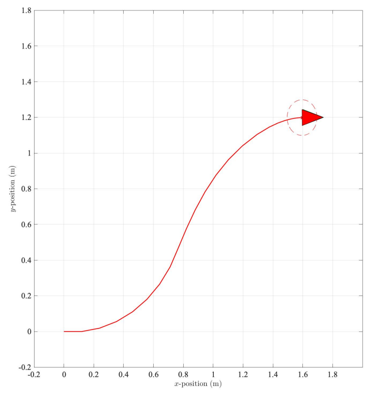

Fig. 6. Multiple shooting with $-0.6 < v < 0.6$ for $\theta_{\text{ref}} = 0$.

Fig. 7 shows robot path taken from predicted control actions when subjected to wider velocity range i.e -1.6m/s $< v <$ 1.6m/s. It has relatively longer trajectory. velocity controls are higher initially (prediction utilizing maximum potential of wider velocity range), trying to minimize x and y components of loss. When bot is close to final point, angular velocity control predictions tries to minimize $||\theta - \theta_{\text{ref}}||^2$ component of loss in order to converge which explains the drastic turn in the trajectory. Convergence is in 32 iterations, error encountered is 0.01. In summary, When velocity constraint is leveraged to take wider range, the robot indeed took more iterations to converge. It is also surprising that error is higher when velocity was relaxed.

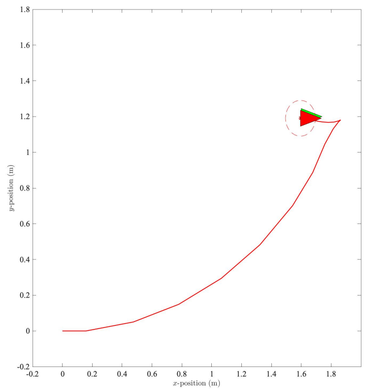

Fig. 7. Multiple shooting with $-1.6 < v < 1.6$ for $theta_{ref} = 0$.

D. *Single and multiple shooting Simulations with Obstacles*

In general, Point stabilization with obstacles requires longer horizon length (N = 50) for convergence as control actions should take care of extra constraint (robot's state should not coincide with obstacle region) in the farther trajectory. Observations from Fig. 8 and Fig. 9, clearly suggests single shooting fails to converge to global optima as gets stuck in local minima because of obstacle and a turn in trajectory cannot be expected as it would increase loss function. Multiple shooting easily converges with less error in-spite of constraint on state. Note that θ_{ref} is $\pi/2$ and velocity range was set between -1.6 m/s to +0.6 m/s.

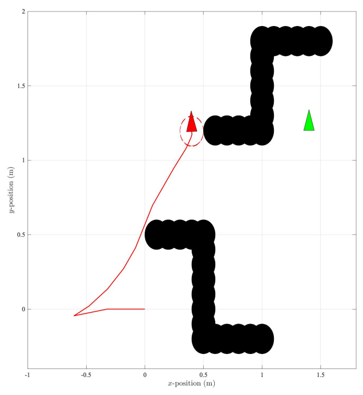

Fig. 8. Single shooting trajectory with obstacle

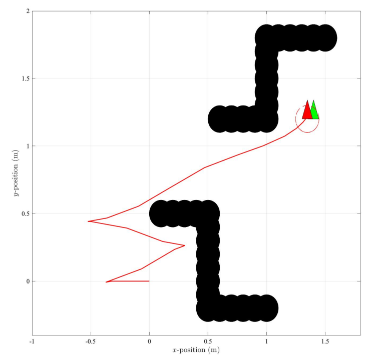

Fig. 9. Multiple shooting trajectory with obstacle

VI. Acknowledgment

We would like to thank Professor Moritz Diehl and Mr. Florian Messerer for their supervision in the project.

References

[1] Mohamed W. Mehrez, ”Optimization Based Solutions for Control and State Estimation in Non-holonomic Mobile Robots: Stability, Distributed Control, and Relative Localization”

[2] T. Faulwasser and R. Findeisen, “Nonlinear model predictive control for constrained output path following,” IEEE Trans. Automat. Control, vol. 61, no. 4, pp. 1026–1039, April 2016.