Dynamic Target Tracking & Collision Avoidance Behaviour of Person Following Robot Using Model Predictive Control*

Avijit Ashe[1] and K. Madhava Krishna[1]

***Abstract*— Executing a robust person-following behaviour is a crucial task for service robots. This requires real-time tracking of the target person and maintaining it in its field of view (FOV) as long as possible while also keeping a safe distance without collision. Such a robot has to autonomously traverse long corridors, bend around corners, curve around an obstacle and join the person back, or even wait them out. Thus, in an unknown urban setting, with other agents and service robots, dynamic target tracking and obstacle avoidance becomes a challenging task for autonomous non-holonomic robots. Model Predictive Control (MPC) has the ability to incorporate future predictions. vehicle kinematics, and non-linearity of constraints while still being reasonably fast for slow and medium walking speeds of a human being. Thus, by solving a single non-linear MPC we can track the unknown dynamic trajectory of a person in real-time, and avoid undesirable proximity.**

In this paper, a constrained MPC framework is designed that meets the above conditions gracefully. The person-following robot (PFR) is able to track a moving target in a simulated environment. With an adequate upper margin of 20Hz, the MPC sends velocity commands for a collision-free trajectory among an increasing number of obstacles in its perceived periphery. In an incremental fashion, it is also able to adapt to varying walking behaviour, that is, types of trajectories and walking speed.

I. INTRODUCTION

Human-robot interaction has a growing impact as assistive or service robot applications are migrating to homes, supermarkets, restaurants, airports, warehouses, and malls. In that segment, a sub-segment of person-following robots (PFRs) have gained a lot of attention from researchers worldwide. This paper deals with developing a robust person following behaviour for a mobile robot. In a dynamic environment with human beings and dynamic obstacles, such as other mobile robots, active collision avoidance is crucial. To solve this problem we need to divide the above task into sub-tasks, and lay out the various methods for solving each of them separately. From the perspective of a mobile robot [1](Flacco et al) this can be segregated into three parts: 1) Environment perception, 2) Collision avoidance, 3) Robot control.

Non-holonomic robot control in an autonomous path-following scenario has been implemented[4] in several different ways. While [3] has implemented a purely laser-based sensor system for the perception part, [5] and [6] have used a combination of stereo cameras and laser sensors, and a RGBD-sensor with a laser scanner, respectively. For the task of collision avoidance, Artificial Potential Field and UKF (Unscented Kalman Filters), Adaptive Window Approach [2], Velocity Obstacle, and Collision Cone have been extensively studied and implemented. Fulgenzi et al [9] devised a method that computed the probability of collision against obstacles in the environment with dynamic uncertainty. However, target tracking [2] with simultaneous collision avoidance is a little explored territory.

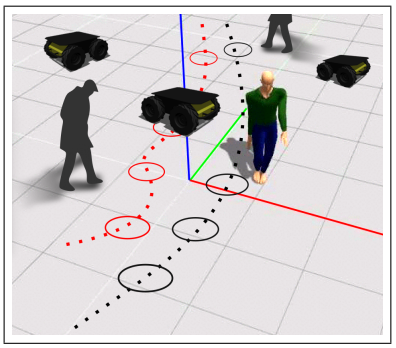


Fig. 1: Typical Person-Following Behaviour

In the field of mobile robot control target tracking is the ability of the robot to update its position, such as (x, y) in a 2D configuration space and orientation according to the target's motion. When a target moves, it traces a trajectory (x, y, t) and the robot's controller has to send velocity commands (v_x, v_y) such that it obeys the necessary constraints. As MPC can incorporate nonlinearity in the objective function as well as the constraints, and use future predictions to update its weights and control parameters to improve the solution, it has been widely used as an advanced method of process control. One must note that MPC is not an algorithm but a methodology to devise your own algorithm for your specific application. MPC can be used adaptively, in a gain-scheduled manner, or in a way like NMPC [7] for nonlinear plant models.

There have been other approaches to solve the control problem (Part 3) solely or in combination with Part 2 (Collision avoidance), about which we talk about in "Related Works" section. To put concisely, the novelty of our work is to solve this problem specifically for a person-following differential-drive robot. We, therefore, 1) Consider an unknown environment with increasing complexity, 2) Perform simultaneous tracking & collision avoidance, and 3) Compute MPC for real-time behaviour.

II. RELATED WORK

In this section, we first review the existing approaches in obstacle avoidance and target tracking, in the lieu of dynamic environments. We introduce person-following robots, and consequently, review the MPC frameworks specific to designing control laws for a robust person-following behaviour.

A. MPC for Obstacle Avoidance & Target Tracking

MPC was first used in motion planning to perform obstacle avoidance for non-holonomic robots by formulating single-

*This work was not supported by any organization

[1]Robotics Research Center, International Institue of Information Technology, Hyderabad, India, 500032

step MPC frameworks using collision cone and velocity obstacle. Then, MPC was integrated with neural networks by Li et al [11] to improve the trajectory tracking behaviour for non-holonomic robots. They formulated a constrained quadratic programming problem that was solved using a primal-dual neural network.

Eslami et al [2] were the first to combine target tracking and collision avoidance using a single MPC framework. It, however, performed it on a predetermined map. To that extent, they formulated a quadratic problem that sends the incremental velocity inputs to the controller using a novel switching action. By carrying out their simulations in Matlab, with a Single-Step MPC, they reported a convergence time of 7-10s for linear velocities and 2-5s for angular velocities. Their tracking error would reduce to 0.1m over 6s and stabilizes in 10s. However, as the number of obstacles was increased, it did poorly.

Tallamraju et al [8] attempted a multi-robot problem that involved tracking each aerial vehicle and avoiding collisions as well. For that, each robot ran a separate local MPC subject to kinematic constraints and robot dynamics. Though it immediately mitigated the problem of local minima with artificial potential fields, it also introduced prolonged convergence time. Moreover, it's extension to non-holonomic robots, as in our case, proved non-trivial.

The latest effort in this direction was from Leman et al [9] where they designed a single nonlinear MPC for path tracking as well as avoiding obstacles on a highway with various speed limit constraints. They developed a framework for a 3DOF vehicle in which the control variable is just the steering angle. With the help of a single MPC, they still manage to do reasonably well at varying speeds. However, they consider static obstacles only and also straight-line trajectories, pretty uncommon for person-following behaviour.

B. MPC for Person-Following Robot

A person-following robot (PFR) is often a wheeled (there can be bipeds and tripeds) mobile robot that executes simultaneous tracking and collision avoidance, while following a person. MPC performs the control (Part 3) for motion planning, assuming Part 1 and 2 are ideal and available. It has a guidance mechanism in form of a person who provides the MPC with a collision-free reference trajectory that it must track, tracing the footsteps. In static environments, it is a passive process, but in dynamic environments tracking the person, maintaining it in its FOV, and avoiding collision with the target and other agents around - must be performed by the controller alone.

The above methods either don't do both together or consider only a very specific scenario such as a straight line on a highway, or without moving obstacles in the scene or predetermined trajectory. A PFR scenario does not have the complete trajectory information beforehand and works in real-time. As the person starts moving farther, the PFR's controller activates and begins to follow. The above MPC formulations do not address such cases.

A PFR presents a less conservative and streamlined application setting for a single-step MPC, with room for passive as well as active obstacle avoidance with reasonable convergence speed. We show here that such a framework is agile and adapts to varying walking characteristics of a person as well.

III. Kinematic Model

In order to design behaviours or controllers for robots we inevitably need models of how they behave and move. First, we begin with the most familiar/basic model used in mobile robotics, which is a differential drive wheeled mobile robot (WMR). We gradually build up from that formulation and tweak it for our intended application and simulation.

A. Model 1.0

An extremely common differential drive robot comprises of two powered wheels and a castor wheel. These two wheels (left and right) can be rotated with different velocities to move the robot around. For example, Amazon's Kiva bots that transport racks of products in their warehouses are modified differential drive two-wheeled robots.

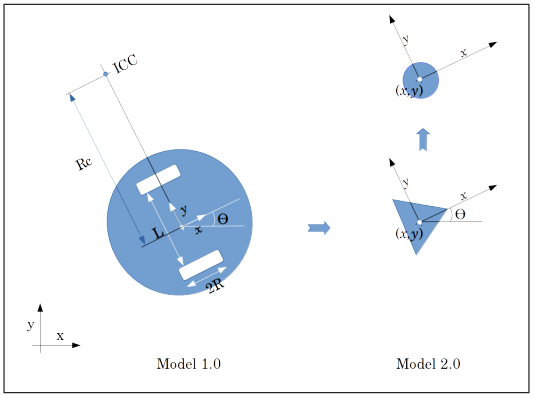


Fig. 2: Kinematic model for mobile robots

A good model is one where you don't have to know what the exact parameters are. For example, the friction coefficient between the wheels and the ground should not be necessary. So, we begin with the distance between the wheels or wheelbase (L), and radius (R) of the wheels, just two. This gives 1.0 using which we want to control v_r, v_l, the forward velocities for right and left wheels respectively, the input signals here, moving about the Instataneous Center of Curvature (ICC).

$$\dot{x} = R/2\big(v_r + v_l\big)cos(\theta)$$
$$\dot{y} = R/2\big(v_r + v_l\big)sin(\theta)$$
$$\dot{\theta} = R/L\big(v_r - v_l\big)$$

where all we want is the location (x,y), and the bearing θ. But, the above differential drive model gives us the rate of change instead, which is very unnatural and unintuitive, to begin with. Hence, we make a small change here and find our next model.

B. Model 2.0

We want a model where we can directly give **u**, the translational velocity or forward velocity of the bot, and ω the angular velocity, as control inputs and drive the bot around. Here, **u** has two components v_x, v_y. This can be done by bringing the **Unicycle** model, Model 2.0, around as follows.

$$\left.\begin{aligned}\dot{x} &= \mathbf{u}cos(\theta)\\ \dot{y} &= \mathbf{u}sin(\theta)\\ \dot{\theta} &= \omega\end{aligned}\right\} \tag{1}$$

Now, Model 1.0 is useful when implementing the controller (on P3DX for example), but for designing the MPC we always use Model 2.0. Here, we shall use this model and gradually build our MPC for a wheeled PFR. The way to connect the control inputs to the state when going from Model 2.0 to Model 1.0 is simply like so.

$$(v_r, v_l) = \left[\frac{2\mathbf{u} + \omega L}{2R}, \frac{2\mathbf{u} - \omega L}{2R}\right] \tag{2}$$

where the RHS (Right Hand Side) are the control inputs that MPC optimizes for generating a collision-free trajectory. That is, it yields a set of v_x, v_y for each time step, t, as it follows the target person.

IV. Optimization

We consider the following optimization problem with the **u(x)** as the control variables, pertaining to the Unicycle-Model of a WMR. The optimized forward linear velocities $\mathbf{u(x)} = \left[v_x(t_i), v_y(t_i)\right]$ have constant acceleration over a time interval $\left[t_i, t_{i+1}\right]$ resulting in a tracking trajectory **x(t)**.

$$\begin{aligned}\underset{v_x, v_y}{\text{minimize}} \quad & (v_x - v_x^{pref})^2 + (v_y - v_y^{pref})^2\\ \text{subject to} \quad & 3(a) : \text{Kinematic Constraints}\\ & 3(b) : \text{FOV Constraints}\\ & 3(c) : \text{Collision Avoidance Constraints}\end{aligned} \tag{3}$$

where, v_x^{pref} and v_y^{pref} are the preferred forward velocities along (x,y) permissible by the physical constraints of the bot. The cost function conforms to minimize the forward velocities penalizing deviations from a fixed set of values, standard protocol in a PFR scenario.

A. Kinematic Constraints

These set of constraints conform to the non-holonomic motion of the bot,

$$\left.\begin{aligned}x_{t+dt} &= x_t + v_x cos(\theta_t)dt\\ y_{t+dt} &= y_t + v_y sin(\theta_t)dt\\ \theta_{t+dt} &= \theta_t + \frac{d\theta}{dt}dt\end{aligned}\right\} \tag{4}$$

which may be relaxed to the following holonomic form for ease, such that $\Delta t = [t_{i+1} - t_i]$

$$\begin{aligned}x_{t+1} &= x_t + v_x \Delta t\\ y_{t+1} &= y_t + v_y \Delta t\end{aligned} \tag{5}$$

where, $\frac{d\theta}{dt} = \omega$, the angular velocity. The limits on these control variables are $-1 \leq v_x \leq 1, -1 \leq v_y \leq 1$, and -2π rad/s $\leq \omega \leq 2\pi$ rad/s.

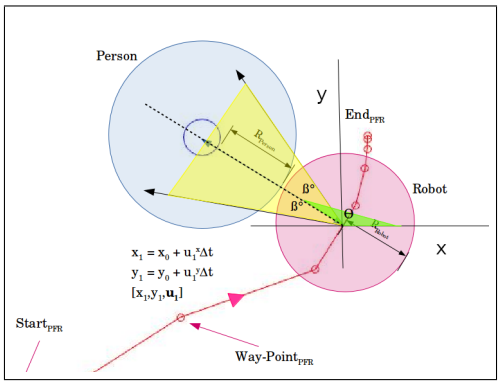


Fig. 3: FOV constraint for an arbitrary angle β

B. FOV Constraints

These set of constraints, illustrated in Fig. 3, conform to the line-of-sight behaviour of the bot. The characteristics of a person following service robot is to ensure the person is in sight mostly all the time, to avoid missing its reference trajectory. For this, we can assume a 360° FOV as a generic case and some arbitrary $\pm\beta°$ a more specific case.

$$b^2 \leq \left(x_{t+dt} - x_{t+dt}^{person}\right)^2 + \left(y_{t+dt} - y_{t+dt}^{person}\right)^2 \leq a^2 \tag{6}$$

for 360° FOV, where a^2 and b^2 are the lower and upper distance bounds on the robot. In other words, it must stay within this ditance limits to not lose sight of the person. Now, if we assume the PFR has an onboard sensor that scans $\alpha°$ on either side of the line joining the current position of the robot and the person, it can be re-written as so,

$$(\theta_{bot} + \beta)^2 \leq \tan^{-1}\left(\frac{y_t^{person} - y_t^{bot}}{x_t^{person} - x_t^{bot}}\right) \leq (\theta_{bot} - \beta)^2$$

where, the global θ from (4) must be added to convert the above angles to world frame.

C. Collision Avoidance Constraints

These sets of constraints conform to avoiding static and dynamic obstacles in the perceived periphery or sensor radius of the PFR. In an ideal case, we assume perfect mapping and availability of their global position information.

$$x_{t+dt} - \begin{pmatrix} x_t^{obs1}\\ x_t^{obs2}\\ \vdots x_t^{obsN}\end{pmatrix} + y_{t+dt} - \begin{pmatrix} y_t^{obs1}\\ y_t^{obs2}\\ \vdots y_t^{obsN}\end{pmatrix} \leq (r_{bot} + r_{clear})^2 \tag{7}$$

where a generic form of circular clearance radius of r_{bot} and r_{clear} is assumed for the PFR and obstacles, respectively. So, any obstacle on the impending course is a sample of points on its periphery, if static, or a sample of points on its trajectory, if dynamic. In case of a PFR, its sensor radius R_{Sensor} is the perceived boundary (if 360° FOV is considered). For other sensor types, the formulations will vary. Regardless, the above set of constraints will be invoked only when an obstacle appears within the perceived boundary of the PFR.

V. RESULTS

The proposed formulation was implemented using the *fmincon* module in Matlab R2017a. The goal here is to study how MPC adapts to varying changes in scenarios with increasing complexity such as types of trajectories taken by the target person and distribution of obstacles, static and dynamic.

A. Standard Notations

A solid BLUE line denotes the reference trajectory generated by the moving target (person) while the RED line denotes the PFR's computed trajectory. The "small circles" on the trajectory in the same color denote hopping vertices or nodes and not the time. That is, within $t = 1$ to $t = 2$, the controller usually sends few sets of control inputs. Similarly, the person can execute several nodes in 1s duration, depending on its walking speed. A BLACK dotted-circle around each blue node on the reference trajectory represents the corresponding node's appropriate collision-free yet FOV-retaining periphery. Bearing markers for the PFR are shown as RED dashed lines, where necessary.

B. Simple Person Following Behaviour

Fig. 4 illustrates an obstacle-free scenario where the objective is to track the person without colliding with it. That is, maintaining the person in its FOV from a safe distance. We assume a 360° FOV in this figure, so do not show the bearing markers.

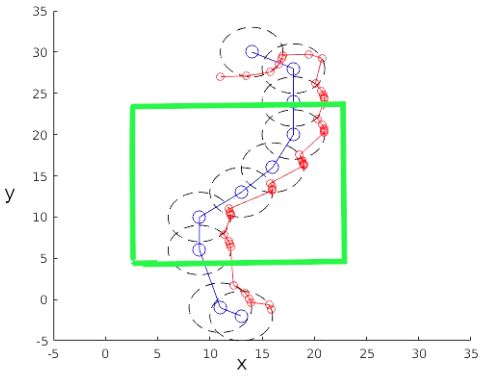


Fig. 4: Simple piece-wise straight line/curvilinear path

We use this first example to explain how the PFR tracks the person from one node to the next one. For this, we focus on the green rectangular region of interest.

It (Fig. 5) shows the transit from $Node0$ to $Node4$ as the person and PFR move in that order, downwards. We select three nodes, $t = k - 1, t = k,$, which is the current node, and$t = k + 1$. As the PFR tracks down, shown by a solid red line, it passes through the circle of $Node1$ towards the periphery of current $Node2$. The MPC uses the (x, y) of this node as the goal and issues the computed (v_x, v_y) for the same. This means that only the boundary of $Node2$, highlighted by solid blue concentric circles, is active, and the other circles disappear clearing the space for free movement. Despite it seems that the red line passes through the circles violating the constraints, it does not, because the other circles (for *Node* $0, 1, 3$) simply cease to exist at $t = k$.

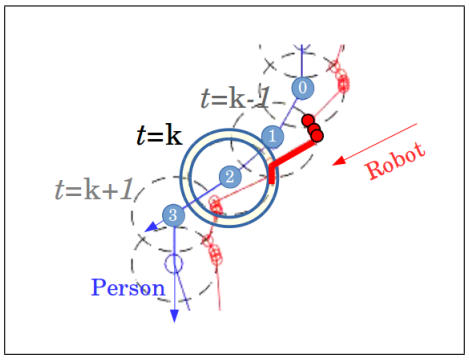


Fig. 5: Node-to-node transition strategy

C. Bending Around Corners

This figure (Fig 6) illustrates a scenario where the person bends around a corner. The multi-colored line segments are walls/obstacles depicting a typical urban indoor setting. While both start roughly at $[0, 0]$, they execute their first right bend together at $[5, 12]$ and the second (left turn) at the exit after crossing the corridor, terminating at $[15, 15]$.

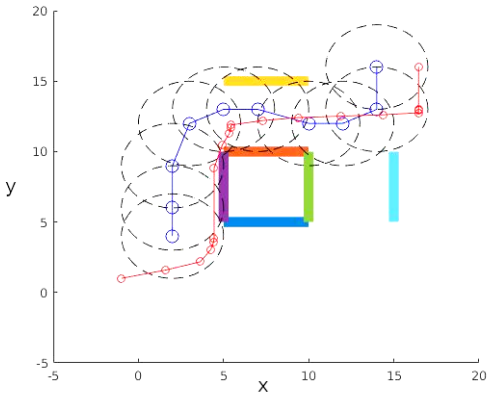


Fig. 6: Indoor setting with walls, a square room, executing two bending around corners

D. Multiple Consecutive Bendings & Corridors

This figure illustrates the ability of MPC to react quickly with a real-time response rate. This agile behaviour demonstrates and loosely mimics a human-like natural person-following characteristics. Here (Fig. 7), we show three walls making up two long corridors. While the path begins coarsely at $[15, 22]$ (top-center) with the bot(red) right behind the person(blue). The PFR moves through the corridor, takes a right turn around the green wall, falls back soon after, and takes another right turn around the purple wall, and continues. As shown, the MPC reacts with agility, waits for some time to let the person complete the turn, and keeps following.

E. Moving Obstacles Crossing Person's Trajectories

In the next scenario, we consider two such situations during a collision course.

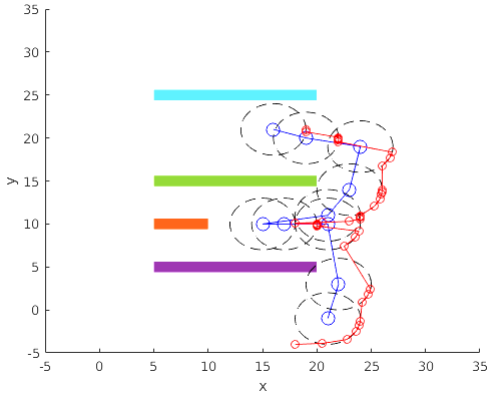


Fig. 7: Multiple long corridors, hallways, executing consecutive turns.

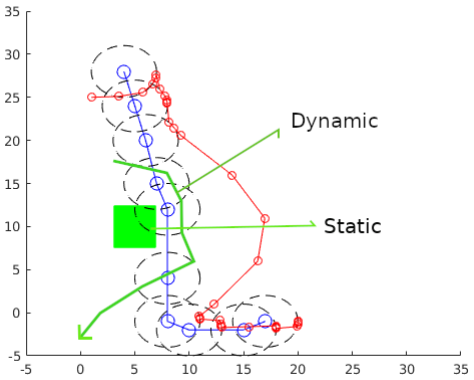
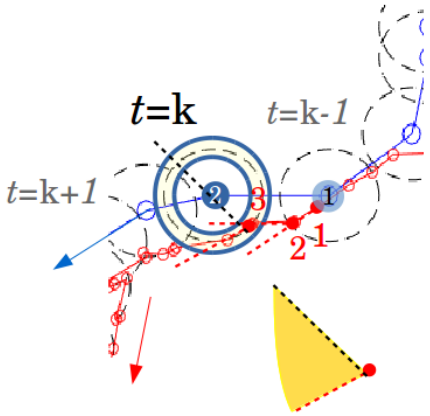
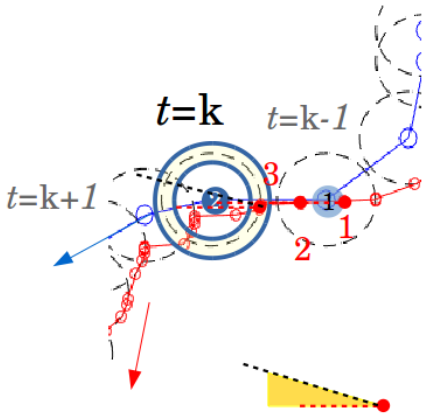


Fig. 8: Scenario with both static and dynamic obstacle



(a) a



(b) b

Fig. 9: Top (a) shows trajectory with 360° FOV, while bottom (b) shows the same with 30° FOV

The above simulation (Fig. 8) illustrates a dynamic obstacle in the form of another person or another PFR in the vicinity. As the obstacle, shown in green trajectory, crosses paths with the target person, the bot waits it out from a safe distance and continues to track the trajectory thereafter. A PFR waits if an immediate solution is unavailable for up to the maximum distance from the person passes or if the person is still. As we can see at $(15, 10)$, when a solution is available, it curves and continues following the target person.

F. FOV Analysis

We show here (Fig. 9) an example of "with and without the FOV constraint", that is, with 360° FOV vs. a smaller arbitrary FOV. We shall see in Section V-G (Fig. 11, Page 6) that despite adding to runtime, it's still agile. In this example, we consider an "S" pattern human movement where all parameters like trajectory length, nodes, number of hopping vertices are kept constant.

With a 360° FOV, bearing markers are unnecessary. Here, we compare the bearings of three PFR nodes as it tracks the trajectory of the person from $Node1$ to 2 from $t = k - 1$ to $t = k$. As $t = k$ is the currently active node (goal (x, y) for the MPC), and the three red dots marked 1, 2 and 3 with bearing markers in dotted-red lines are the PFR's tracks, we focus on them. We see something noticeably different between these positions in the top and the bottom figures.

The difference between the angle of the red bearing marker in $Node3$ of PFR and the black bearing marker (desired angle for direct line of sight) is greater than $\pm 15°$ (Fig. 9a). So, we can verify that the FOV constraint was not applied here. Hence, it stays in a safe distance but with arbitrary bearing. However, in Fig. 9b we can see that the bearing marker of the $Node3$ of PFR makes a very small angle with the line of sight of $Node2$ of the person, confirming that the FOV constraint was applied here. That is, from a set of permissible (v_x, v_y) the MPC chooses a pair that not only tracks the person but also adjusts its bearing to force the line of sight within tolerance ($\pm 15°$ in this case).

G. Quantitative Analysis

When designing a controller, the runtime is an important criterion. A faster runtime means quicker response and agile person-following behaviour. Each time MPC computes the control variables, it does so in a finite number of iterations and time. During a path-tracking trajectory, there are several such computes, also known as planning and replanning. Below (Fig. 10) on Page 6 we show the time consumed per MPC computation during a curvilinear (S-pattern) path-tracking scenario. There can be several other types of human

movement patterns as well, however, which we shall soon see.

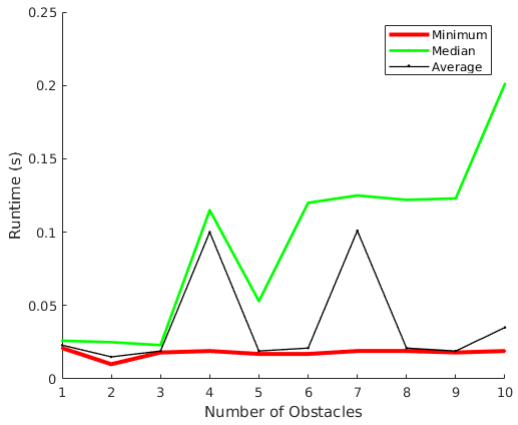


Fig. 10: Number of Obstacles vs Runtime per MPC Computation

We repeat the simulations 10 times for each obstacle scenario. The minimum time per plan (compute) for different number of obstacles average out at around 0.025s. While, the trend is noticeable by looking at the median values, which is a statistical measure dormant to outliers, the average runtime is skewed in this figure. Because 4 and 7 obstacle-scenarios seems to be outliers, that pushes the mean out of the trend, while the rest of the values show a steady-state rise. The median shows that as the number of obstacles increases, there is an increase in runtime as well. It increases from 0.02s to 0.2s, a 10x increase.

In another case, a set of different human movement patterns were considered. They were divided into 6 types and each scene was repeated (non-identically) with a fixed number of hopping vertices and obstacles 10 different times. So, a total of 60 simulations were performed. Then, the average minimum (& maximum) runtime per compute for each trajectory type was recorded.

Human Movement Pattern (10 Hops)	Avg. Min Runtime (s) (10 Rounds) 360º FOV	Avg. Max Runtime (s) (10 Rounds) 360º FOV	Avg. Median Runtime (s) 10 Rounds ±15º FOV	Avg. Median Runtime (s) 10 Rounds ±45ºFOV
	0.010	0.013	0.017	0.015
	0.014	0.018	0.037	0.027
	0.019	0.021	0.030	0.024
	0.011	0.015	0.018	0.013
	0.017	0.120	0.032	0.019
	0.021	0.048	0.037	0.027

Fig. 11: Human Movement Patterns vs Trajectory Tracking Runtime

The chart (Fig. 11) confirms that with a significant increase in complexity there is a noticeable increase in the per compute run-time. For example, 0.018s to 0.021s for the ”S” pattern. This translates to a replanning frequency of 50Hz. Secondly, the runtime for a specific type of trajectory fluctuates between a range too. For example, the ”V” pattern takes between 0.011s and 0.015s per MPC plan during the entire trajectory. This translates to a replanning frequency $\simeq 100$ Hz.

For a specific trajectory type, the runtime for 360° is usually lower while the more constrained 30°or $\pm\,15^\circ$ FOV is the highest, though for easier trajectories the difference is not significant. For example, the ”S” pattern has a median value of 0.019s for 360° FOV, surpassed by 0.024s for 90° FOV, even surpassed by 0.030s for 30°.

To summarize, there are several other factors that can affect the results, such as number of hopping vertices, distance between them, and so on. For the above metric, they were kept pretty much constant. Overall, the MPC performed with an upper margin of 0.05s per plan, resulting in a replanning frequency of 20Hz, across trajectories.

H. Ablations

To identify the violation of constraints with changes in the complexity of scenarios, we consider the following chart (Fig. 12). The trajectory, as in number of nodes, length, and the type of trajectory, is kept constant. To ensure that the obstacles interact with the same, a small area is selected and a random generator is used to generate an increasing number of static (and dynamic) obstacles in the scene. Then, the number of collision avoidance (CA) & FOV violations are recorded along with total number of constraints, in each case.

No Violated FOV (±15º) Constraints	0/0	0/0	0/1	1/3	1/3	3/4
No Violated Collision Constraints	0/0	3/5	5/7	9/10	11/13	13/16
Interacting Static/Dynamic Obst	0/0	1/1	2/2	3/3	4/4	5/5
Total Number Of Constraints	42x3	84x8	126x9	168x7	210x12	252x18

Fig. 12: Violation of CA and FOV constraints vs Scene Complexity

In Fig. 12, the first row shows the number of violated FOV constraints, the second row shows the number of violated collision avoidance constraints, the third row shows the variation in number of static/dynamic obstacles, and the fourth row shows the total number of constraints. The last value is computed as the *number of collision constraints in the entire trajectory* times *the avg. number of iterations per MPC compute*. In Fig. 13 we show the violation of FOV and collision constraints with the change in trajectory type, for a fixed distribution of obstacles.

No Violated FOV (±15º) Constraints	0	0	1	1	2	4
No Violated Collision Constraints	0	0	3	7	6	10
Human Movement Pattern						

Fig. 13: Violation of CA and FOV constraints vs Movement Patterns

To ensure that the baseline is clean, we consider a scenario without zero constraint violations. We then introduce obstacles in the scenario such that it interacts with the target person's path and records the changes. As the number of obstacles increase, so does the number of constraints. From these (Fig. 12, 13) we can infer a linearly rising trend in both FOV and CA constraints. However, the FOV constraint appears to be minutely affected by the variation. For example, in Fig. 12 Col3-Row3, we have 5 CA violations for 2 static obstacles. For 2 dynamic obstacles, we get 7 CA violations. In contrast, we only get 0 and 1 FOV violations for static/dynamic respectively.

VI. CONCLUSIONS

In brief, we have approached the problem of person-following mobile robot in an urban setting using a 2D simulated environment. In this paper, we derive the proposed MPC formulation, look at specific constraints that impose walking behaviour traits of a person, such as different trajectories, and distribution of obstacles. To have a fully functional social robot, we have to assist the human-centric goal-reaching behaviour with collision avoidance with obstacles in the environment as well as with the person itself. The final achievements can be summarized as below.

- The formulation solves a single MPC to perform tracking as well as collision avoidance.
- The MPC works reasonably fast with an upper margin of 20Hz across trajectories, which is a lot more than a practical person walking speed.

FUTURE WORK

The design of control law for a mobile robot depends on its motion or mechanical behaviour. And, understanding this behaviour starts with understanding the wheel constraints placed on its mobility. A person following robot at a mall will have a different locomotion type (mobility constraints) from that at an airport, a warehouse, a restroom, unpaved roads, or a crosswalk. So, the focus can also be on developing MPCs for each of these specific applications of social robots.

In practical scenarios such as implementing this MPC on a P3DX via ROS, we have to solve the inverse kinematics problem where given the pose $[x, y$or $\theta]$, we have to derive the steering angle (β) and left-right wheel speeds (Φ), as a function of time (t).

Eventually, the long-term goal is obviously to add predictability and out-of-sight tracking to make the person following behaviour as natural as possible.

ACKNOWLEDGMENT

The authors would like to acknowledge Mithun Babu Nallana (mithun.babu.n@gmail.com) from IRL Lab, RRC, IIIT Hyderabad for his prompt suggestions and directions.

REFERENCES

[1] Flacco, Fabrizio, et al. "A depth space approach to human-robot collision avoidance." 2012 IEEE International Conference on Robotics and Automation. IEEE, 2012.

[2] Eslami, Narges, and Roya Amiadifard. "Moving Target Tracking and Obstacle Avoidance for a Mobile Robot Using MPC." 2019 27th Iranian Conference on Electrical Engineering (ICEE). IEEE, 2019.

[3] Cen, Minfeng, et al. "Real-time Obstacle Avoidance and Person Following Based on Adaptive Window Approach." 2019 IEEE International Conference on Mechatronics and Automation (ICMA). IEEE, 2019.

[4] Y. Gao, C. G. Lee, K. T. Chong, "Receding horizon tracking control for wheeled mobile robots with time-delay", Journal of mechanical science and technology, vol. 22, pp. 2403-2416, 2008.

[5] Tsalatsanis, Athanasios, K. Valavanis, and Ali Yalcin. "Vision based target tracking and collision avoidance for mobile robots." Journal of Intelligent and Robotic Systems 48.2 (2007): 285-304.

[6] Kim, Minkyu, et al. "An architecture for person-following using active target search." arXiv preprint arXiv:1809.08793 (2018).

[7] Kehler, Johannes, Matthias A. Mller, and Frank Allgwer. "A nonlinear model predictive control framework using reference generic terminal ingredients." IEEE Transactions on Automatic Control (2019).

[8] Ge, Shuzhi Sam, and Yun J. Cui. "Dynamic motion planning for mobile robots using potential field method." Autonomous robots 13.3 (2002): 207-222.

[9] Fulgenzi, Chiara, Anne Spalanzani, and Christian Laugier. "Dynamic obstacle avoidance in uncertain environment combining PVOs and occupancy grid." Proceedings 2007 IEEE International Conference on Robotics and Automation. IEEE, 2007.

[10] K. Benbouabdallah and Z. Qi-dan, "Design of a fuzzy logic controller for a mobile robot tracking a moving target," in Computer Science and Network Technology (ICCSNT), 2012 2nd International Conference on, pp. 634-638, 2012

[11] Z. Li, J. Deng, R. Lu, Y. Xu, J. Bai, and C.-Y. Su, "Trajectory-tracking control of mobile robot systems incorporating neural-dynamic optimized model predictive approach," IEEE Transactions on Systems, Man, and Cybernetics: Systems, vol. 46, pp. 740-749, 2016.

[12] Tallamraju, Rahul, et al. "Decentralized mpc based obstacle avoidance for multi-robot target tracking scenarios." 2018 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR). IEEE, 2018.

[13] Leman, Zulkarnain Ali, et al. "Model Predictive Controller for Path Tracking and Obstacle Avoidance Manoeuvre on Autonomous Vehicle." 2019 12th Asian Control Conference (ASCC). IEEE, 2019.