2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

# A Comprehensive Trajectory Planner for a Person-Following ATV

Huckleberry Febbo, Jiawei Huang, and David Isele

*Abstract*— **This paper presents a trajectory planning algorithm for person following that is more comprehensive than existing algorithms. This algorithm is tailored for a front-wheel-steered vehicle, is designed to follow a person while avoiding collisions with both static and moving obstacles, simultaneously optimizing speed and steering, and minimizing control effort. This algorithm uses nonlinear model predictive control, where the underling trajectory optimization problem is approximated using a simultaneous method. Results collected in an unknown environment show that the proposed planning algorithm works well with a perception algorithm to follow a person in uneven grass near obstacles and over ditches and curbs, and on asphalt over train-tracks and near buildings and cars. Overall, the results indicate that the proposed algorithm can safely follow a person in unknown, dynamic environments.**

## I. Introduction

An important use-case for robotic platforms is transporting heavy or otherwise difficult or dangerous payloads. Using a person following algorithm as a control mechanism enables an operator to intuitively guide the robot, hands-free, as they themselves navigate through unknown dynamic environments. To satisfy the terrain and payload requirements, we select an autonomous all-terrain vehicle (AATV) as our robotic platform (see Figure 1). To ensure the safe and efficient behavior of this platform in unknown, dynamic environments, the person following algorithm should have all of the specifications listed in Table I and run in real-time.

Person following has been researched on a wide variety of robotic platforms [1], including wheelchairs [2,3], unmanned aerial vehicles [4], legged robots [5], and skid-steer platforms [6]. While many of the person following algorithms tailored for these platforms include subsets of the specifications in Table I, an algorithm that includes all of them does not currently exist.

The core functionality we desire is a person following behavior ($S_1$, Table I), but is not, by itself, sufficient for safety in most realistic environments. There are examples that demonstrate the effectiveness of PID controllers and other tracking algorithms in relatively controlled environments like office buildings [7], but our use-cases demand an ability to operate in locations with clutter and even moving obstacles ($S_2$, Table I). There is a need to handle moving obstacles, because people or other vehicles may cut in between the operator and the robot — assuming a static environment would result in a miscalculation of the available open space and potentially even a collision.

For safety, in addition to obstacle avoidance, it is also necessary for our planning algorithm to be suitable for a front-wheel steered vehicle ($S_3$, Table I), like the ATV platform selected for this research. Differential-drive vehicles are by far the most studied vehicle model for person following [2, 3, 7–9]. However, unlike differential-drive vehicles, front-wheel-steered vehicles do not have an ability to turn in place and are instead bound by stricter, non-holonomic constraints. These more complicated kinematics limit the maneuverability of the robot, making it more difficult to follow the person's trajectory. Together, these considerations motivate the need to generate vehicle-specific trajectories.

Fig. 1: **Autonomous all-terrain vehicle (AATV)** This platform was used to collect all of the experimental results provided in this paper.

Developing a planning algorithm with $S_1$-$S_3$ would be a step forward for person following algorithms. However, the behavior of the vehicle controlled with such an algorithm may not be safe or efficient. For safety, the vehicle may unnecessarily collide with an obstacle if speed and steering are not simultaneously optimized ($S_4$, Table I). When speed and steering are simultaneously optimized, the system can reduce its speed to make tighter turns and is therefore more responsive to obstacles [10, 11]. For efficiency, the control system is likely to cause the vehicle to use excessive fuel and induce excessive mechanical wear if the planning algorithm does not incorporate these considerations, which can be accomplished by including a term that minimizes the control

TABLE I: Planner Specifications

| Specification | Description |
|---|---|
| $S_1$ | person following behavior |
| $S_2$ | static and moving obstacle avoidance |
| $S_3$ | suitable for a front-wheel steered vehicle |
| $S_4$ | optimization of speed and steering |
| $S_5$ | minimum control effort |

978-1-7281-6211-9/20/$31.00 ©2020 IEEE

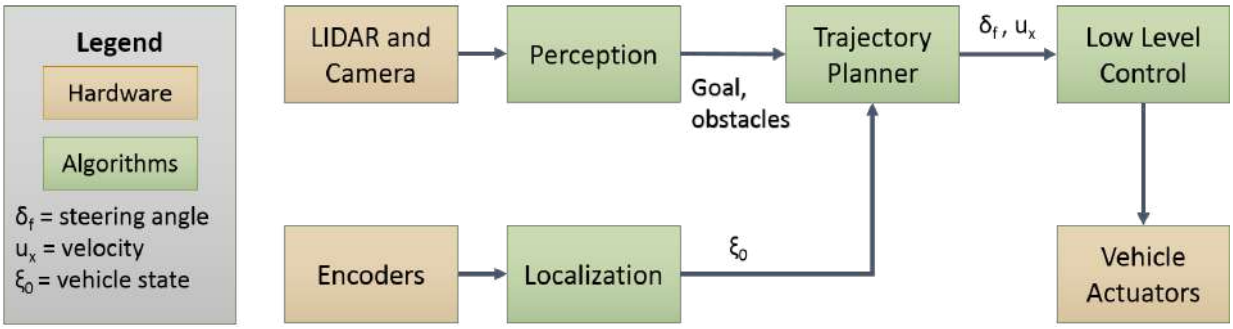



Fig. 2: **System Architecture.** A diagram of information flow through our system.

effort ($S_5$, Table I) [12–15].

The ultimate goal of this research is to develop a trajectory planning algorithm with all of the specifications listed in Table I. This paper uses a nonlinear model predictive control (NMPC)-based trajectory planner, which is a popular approach for developing comprehensive trajectory planners [10, 10, 13–17].

This paper makes the following contributions:

- Introduce an NMPC-based person following algorithm with $S_1$-$S_5$.
- Evaluate the limits of the developed algorithm to control the AATV in Figure 1 in a known, dynamic environment.
- Evaluate the developed algorithm to control the AATV in an unknown environment.

The controller is one component of a larger system. This paper starts by describing the system architecture (Section II) to frame the context of the controller. Then, in Section III, the mathematical formulation is described. Section IV describes the experimental procedure used to evaluate the system, and Section V discusses the results. Finally, Section VI provides the conclusion.

## II. System Architecture

The main contribution of this paper is a novel trajectory planning algorithm, which is described in detail in Section III. However, to enable testing of this planning algorithm in an unknown environment, it must be combined with perception, localization, and low level control algorithms, as depicted in Figure 2. Perception algorithms process data from sensors to generate estimates for the obstacles in the environment and the goal (i.e., the person). At the same time, the localization algorithm uses data from the steering and speed encoders to estimate the vehicles state using an odometry-based estimate. Odometry-based estimation of the state is subject to drift in the ($x$, $y$) position of the vehicle, but since the optimal control problem is formulated in local coordinates, this simplified estimation does not affect safety and performance. Together, the output from both the perception and localization algorithms establish the data needed to solve the trajectory planning problem. The output of the trajectory planner is the desired steering angle and speed, which gets fed to low-level PID controllers. These controllers are responsible for controlling the voltage of the vehicle's steering and throttle actuators.

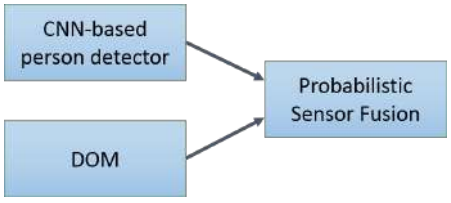



Fig. 3: **Perception Architecture.** Perception combines camera-based person detection with a LIDAR-based dynamic occupancy map.

While, the localization and low-level algorithms are well established methods, the perception algorithm is novel and merits a more thorough treatment. The remainder of this section gives an overview of the perception algorithm that was developed to enable operation of the AATV in an unknown environment[1].

### A. Perception

The task of the perception module is to (1) accurately estimate the position and velocity of the person to be followed (e.g., operator) and (2) detect and determine the location and shape of any obstacle in the robot's surrounding area. To achieve these tasks, a top-mounted LIDAR, a front-facing LIDAR, and a front-facing monocular camera are used. Section IV describes these sensors in greater detail, this section provides the details of the algorithms that process the data generated from these sensors.

Figure 3 illustrates the architecture of the perception subsystem; objects of interest are detected using two independent modules. The first module is a camera-based person detector that detects any pedestrian in the camera's field of view. This detector is a region-proposal based object detector, which has a similar structure to Faster-RCNN [18]. We trained the detector on a proprietary data-set. Images in the dataset were hand-labelled, and collected from diverse sources. The detector's output consists of bounding box locations of detected pedestrians as well as a rough estimated distance to these pedestrians The second module is the

[1]A full analysis and evaluation of the perception algorithm will be the subject of a future publication.

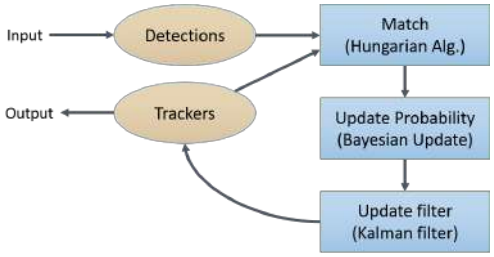



Fig. 4: **Sensor Fusion Architecture.** Diagram of the sensor fusion pipeline.

Dynamic Occupancy Map (DOM) [19], which uses the two LIDAR sensors on the robot to detect any obstacle within a certain distance to the robot. LIDAR scans are filtered to a max distance of $25$ m and obstacles are defined as any object with a height of over $0.7$ m above ground. The output of DOM is a list of convex hulls and each convex hull describes an obstacle's spatial dimensions. The outputs of the person detector and DOM cannot be directly utilized by the controller because camera detection lacks position accuracy while DOM lacks object category. Therefore, a module is needed to fuse the two detection results. In addition, since missed or false detections are always possible, the fusion module should provide some tolerance to noisy detection inputs.

The fusion module's architecture is shown in Figure 4. We first perform a matching between incoming detections and existing trackers using bipartite matching (the Hungarian algorithm). For this algorithm to function, we define a cost function between a detection and tracker as follows

*a) LIDAR detections:* The Euclidean distance between detection and tracker centers as the cost.

*b) Camera detections:* The pixel distance between the projection of the tracker onto the image plane and bounding box center as the cost. The matching yields three types of outcomes. For a matched detection and tracker, we use the detection to update the tracker. For unmatched trackers, we update them with negative (e.g., empty) detection. For unmatched detections, we allow them to generate new trackers.

In order to seamlessly fuse the two types of detections, we choose to model the existence probability $P_{exist}$ of each tracked object. Regardless of the source of detections, each positive (e.g., matched) detection increases the probability of the tracker, while each negative detection decreases it. By applying Bayes' Rule, we can explicitly calculate the existence probability from inverse sensor model, $P(\text{existence} \mid \text{measurement})$. We adopt a simple inverse sensor model by assuming certain false positive and false negative rates for each detection module. For objects outside a sensor's field of view, their $P_{exist}$ does not change.

The $P_{exist}$ is used to create new trackers and delete obsolete trackers. A tracker is created whenever its $P_{exist}$ exceeds a particular high threshold. This tracker is then deleted when its $P_{exist}$ drops below a particular low threshold. Then, the position and velocity of every pedestrian is estimated using a Kalman filter with a constant velocity model.

It is noted that, for initial person acquisition, the person must be detected simultaneously by the camera and the LIDAR. Afterward, the person continues to be tracked, even if they leave the field of view of one of these sensors.

## III. Trajectory Planner

At the heart of and NMPC problem lies an trajectory optimization problem (e.g., optimal control problem). This section describes the optimal control problem that is formulated in this work to incorporate the planner specifications listed in Table I. This entire formulation fits into a single-phase, continuous-time, optimal control problem in a Bolza form [20] tailored for NMPC given in Eqn. 1 - Eqn. 8 as

$$\underset{\xi(t),\zeta(t),\xi_{\mathbf{0}_s},\xi_{\mathbf{f}_s},\, t_f}{\textbf{minimize}} \quad J = \mathcal{M}(\xi(t_0), t_0, \xi(t_f), t_f) + \int_{t_0}^{t_f} L(\xi(t), \zeta(t), t)\, dt + \mathbf{w}_{s0}\xi_{\mathbf{0}_s} + \mathbf{w}_{sf}\xi_{\mathbf{f}_s} \tag{1}$$

$$\textbf{subject to} \quad \frac{d\xi}{dt}(t) - f(\xi(t), \zeta(t), t) = 0 \tag{2}$$

$$C(\xi(t), \zeta(t), t) \leq \mathbf{0} \tag{3}$$

$$\xi_{\mathbf{0}} - \xi_{\mathbf{0}tol} \leq \xi(t_0) \leq \xi_{\mathbf{0}} + \xi_{\mathbf{0}tol} \tag{4}$$

$$\xi_{\mathbf{f}} - \xi_{\mathbf{f}tol} \leq \xi(t_f) \leq \xi_{\mathbf{f}} + \xi_{\mathbf{f}tol} \tag{5}$$

$$\xi_{min} \leq \xi(t) \leq \xi_{max} \tag{6}$$

$$\zeta_{min} \leq \zeta(t) \leq \zeta_{max} \tag{7}$$

$$t_{f_{min}} \leq t_f \leq t_{f_{max}} \tag{8}$$

where $t_0$ is the constant initial time, $t_f$ is the variable final time, $\xi(t) \in \mathbb{R}^{n_{st}}$ is the state and $n_{st}$ is the number of states $\zeta(t) \in \mathbb{R}^{n_{ctr}}$ is the control and $n_{ctr}$ is the number of controls, $\xi_{\mathbf{0}}$ is the desired initial state vector, $\xi_{\mathbf{0}tol}$ is the initial state tolerance vector, $\xi_{\mathbf{f}}$ is the desired final state vector, $\xi_{\mathbf{f}tol}$ is the final state tolerance vector, $\xi_{\mathbf{0_s}}$ and $\xi_{\mathbf{f_s}}$ are arrays of slack variables on the initial and terminal states, respectively, and $\mathbf{w_{s0}}$ and $\mathbf{w_{sf}}$ are arrays of weights. The remainder of this section describes the details of Eqn. 1, Eqn. 2, and Eqn. 3.

### A. Cost Functional

First, the cost functional in Eqn. 1 is given in Eqn. 9 as

$$\begin{aligned} J = {} & w_t t_f \\ & + w_g \frac{\Big(x(t_f) - x_g\Big)^2 + \Big(y(t_f) - y_g\Big)^2}{\Big(x(t_0) - x_g\Big)^2 + \Big(y(t_0) - y_g\Big)^2 + \epsilon} \\ & + w_{haf} \int_{t_0}^{t_f} [\sin(\psi_g)(x - x_g) - \cos(\psi_g)(y - y_g)]^2 \mathrm{d}t \\ & + w_{ce} \int_{t_0}^{t_f} [w_{\delta_f} \delta_f(t)^2 + w_{sr} sr(t)^2 + w_{a_x} a_x(t)^2] \mathrm{d}t \\ & + w_{u_x} \int_{t_0}^{t_f} \frac{1}{u_x(t)^2 + \epsilon} \mathrm{d}t \\ & + \mathbf{w}_{s0}\xi_{\mathbf{0}_s} + \mathbf{w}_{sf}\xi_{\mathbf{f}_s} \end{aligned} \tag{9}$$

where, $w_t, w_g, w_{haf}, w_{ce}, w_{\delta_f}, w_\gamma, w_{a_x}, w_{u_x}, \mathbf{w}_{s0}, \mathbf{w}_{sf}$ are constant weight terms, $x(t)$ and $y(t)$ are the vehicle's position coordinates, $x_g$ and $y_g$ are the goal's coordinates, $\epsilon$ is a small number set to $0.01$ to avoid singularities, $\psi_g$ is the desired final heading angle, $\delta_f(t)$ and $sr(t)$ steering angle and rate, respectively, $u_x(t)$ is the longitudinal speed, and $a_x(t)$ is the longitudinal acceleration.

There are six terms in Eqn. 9. The first term minimizes the final time, which is commonly seen in racing applications [21,22], where speed is a primary objective. However, even in a person following system, if the vehicle travels too slowly, then the system performance will be poor. Additionally, if the planner is updating quickly and the changes in the control signals are very small during the execution horizon, the vehicle may not move. Thus, adding a minimum final time term makes the planner calculate more aggressive trajectories, which make the vehicle likely to move. The second term pushes the final position of the vehicle closer to the goal. The third term helps the vehicle travel toward the goal in a particular orientation. To accomplish this, the area between a line passing through the goal point $x_g, y_g$ at the desired goal angle $\psi_g$ and the vehicle's $x, y$ trajectory is minimized [10]. The fourth term minimizes the control effort and the fifth term maximizes speed, which along with minimizing the final time, helps ensure that the vehicle moves. Finally, the sixth term adds slack constraints on the initial and terminal conditions to help avoid finding solutions that are nearly infeasible [23].

### B. Dynamic constraints

The vehicle is modeled using a nonlinear kinematic ground vehicle model [24,25]. The kinematic model is used because, the vehicle moves at slow speeds, so vehicle dynamics are not expected to significantly affect the vehicle's motion. Also, comparisons between the kinematic and the dynamic bicycle model [26] provide additional support for the decision to chose the kinematic model for automated driving algorithms over the dynamic model. In these comparisons, the kinematic model performs similarly well to a dynamic model, while being less computationally expensive [26]. The dynamic constraints in Eqn. 2 are defined using the kinematic vehicle in Eqn. 10 as

$$\begin{aligned}
\dot{x}(t) &= u_x(t)\cos\left(\psi(t) + \tan\left(\frac{l_a\tan(\delta_f(t))}{l_a+l_b}\right)^{-1}\right)\\
\dot{y}(t) &= u_x(t)\sin\left(\psi(t) + \tan\left(\frac{l_a\tan(\delta_f(t))}{l_a+l_b}\right)^{-1}\right)\\
\dot{\psi}(t) &= \frac{u_x(t)\sin\left(\tan\left(\frac{l_a\tan(\delta_f(t))}{l_a+l_b}\right)^{-1}\right)}{l_b}\\
\dot{u}_x(t) &= a_x(t)
\end{aligned} \tag{10}$$

where, $\psi(t)$ is the yaw angle, $l_a = 0.6$ m and $l_b = 0.6$ m are the wheelbase distances.

### C. Path constraints

To avoid collisions with static and dynamic obstacles, time-varying hard constraints on the vehicles trajectory are used to ensure that the vehicle's planned trajectory does not intersect with the obstacles' predicted trajectories [16, 27]; the path constraints in Eqn. 3 are given in Eqn. 11 as

$$\begin{aligned}
&\left(\frac{x(t) - (\mathbf{x0_{obs}}[i] + \mathbf{v_x}t)}{\mathbf{a_{obs}}[i] + sm(t)}\right)^2 +\\
&\left(\frac{y(t) - (\mathbf{y0_{obs}}[i] + \mathbf{v_y}t)}{\mathbf{b_{obs}}[i] + sm(t)}\right)^2 > 1, \text{ for } i \in 1:Q
\end{aligned} \tag{11}$$

where $sm(t) = 0.45 + \frac{0.7-0.45}{t_f}t$ describes the time-varying safety margin. The parameters in $sm(t)$ are chosen based off of the size of the vehicle then tuned as needed. $\mathbf{x0_{obs}}[i]$ and $\mathbf{x0_{obs}}[i]$ describe the position of the center of the $i$th obstacle at time $t$, $\mathbf{a_{obs}}$ and $\mathbf{b_{obs}}$ are arrays of semi-major and semi-minor obstacles' axes, and $Q$ is the number of obstacles.

## IV. Experiment Description

This section describes the experimental platform (Figure 1), the conditions under which the tests are performed, and the software configuration that the tests are conducted with.

### A. Experimental Platform

Our robotic platform is a Honda Foreman Rubicon that is modified to receive autonomous command messages over a CAN bus. For perception, there is a front mounted ZED stereo camera, and 2 Velodyne LIDARSs mounted on the top (HDL-32) and front (VLP-16). All results in this paper are produced using an MSI $GS65$ Stealth $666$ laptop computer running Ubuntu $16.04$ and ROS Kinetic with an Intel Core $\text{i}9-9880H$ CPU @$2.3\text{GHz}\times 16$, and $31.3$GB of RAM.

*a) Known Environment:* In a known environment, both the goal and obstacle data are simulated and provided directly to the planner. Thus, the perception problem is eliminated when the environment is assumed to be known. The purpose of the tests conducted in this environment is to evaluate the bounds of the system's maneuverability.

The right-hand trace of Figure 6 depicts the parametric scenario that is used to collect all the results in the known environment. In this scenario, a goal is placed $20$ m in front of the vehicle and travels straight ahead at a speed of $1$ $\frac{\text{m}}{\text{s}}$ for $25$ s. So, the final position $x$ of goal is $45$ m. This goal has a radius of $4$ m, which is green in color and surrounds an obstacle (i.e., simulated operator) with a radius of $0.75$ m, which is red in color. The position of the goal at various time-stamped positions can be seen in Figure 6. Additionally, there is an obstacle, denoted as $O_1$, moving directly towards the vehicle starting from an $x$ position of $25$ m. This scenario is parameterized by the radius (i.e., size) and speed of $O_1$.

*b) Unknown Environment:* In the unknown environment, the goal and obstacle data need to be estimated. To make this estimation, the data collected from the perception sensors (i.e., ZED camera, top LIDAR, and front LIDAR) is processed using the perception algorithm described in

Section II-A to enable estimation of the goal and obstacle data. Fourteen tests are conducted in an unknown off-road environment, where the vehicle must avoid collisions with obstacles, drive through ditches, and operate on uneven terrain. The tests start when the person is either beside or in front of the vehicle. During the tests, the person walks to random locations and randomly stops several times.

### B. Software Configuration

The literature shows that it is difficult to solve comprehensive trajectory planning algorithms fast enough for various on-line applications [10, 16, 28, 29]. Fortunately, together, the recent advances in programming languages [30], optimization modeling languages [31], automatic differentiation tools [32, 33], and optimal control software [34] make it possible to solve comprehensive problems in real-time. For instance [11] uses these advances to shown that the trajectory optimization problems developed in [10, 16] can be solved in real-time[2]. In this paper `NLOptControl` 0.1.6+ [34] is used in conjunction with the trapezoidal method [35] and the *KNITRO* 10.3 NLP solver. For safety, if the solution to the NLP problem is not optimal then the vehicle is stopped, which is the case when the vehicle travels too closely to an obstacle.

## V. Results

### A. Moving Obstacle Avoidance in a Known Environment

Figure 5 shows that the vehicle attains the goal in sixteen out of the twenty tests conducted in the known, dynamic environment. The speed and the size of the obstacle moving towards the vehicle realize each of these tests, where a faster and larger obstacle is harder to avoid. For the cases tested, $1.83\ \frac{\text{m}}{\text{s}}$ is the fastest obstacle our system can handle, where the obstacle has a radius of $0.28$ m. On the other hand, to avoid an obstacle with a radius of $0.78$ m, the speed is reduced to $1.48\ \frac{\text{m}}{\text{s}}$. We will now consider in detail two of these twenty tests; the first being a successful trial and the second being a failure, where there vehicle crashes into the obstacle.

Figure 6 shows a successful trial selected from the collection of tests plotted in Figure 5. As seen in the right hand trace, the vehicle moves out of the way of an obstacle that has a radius of $0.68$ m and is moving directly toward it at $1.2\ \frac{\text{m}}{\text{s}}$. After safely avoiding this obstacle, the vehicle continues to follow the goal until it is eventually attained at $28.5$ s. In the top left hand trace, it is seen that the planning algorithm is solved in real-time, with an average solve-time of $0.107$ s. In the bottom two traces on the left, it can be seen that the vehicle has relatively smooth trajectories for both steering and speed.

Figure 7a shows one of the four cases in Figure 5 where the vehicle is not able to avoid a collision with the obstacle. In this case, where obstacle's radius is $0.39$ m and its speed is $1.7\ \frac{\text{m}}{\text{s}}$, the vehicle does not move quickly enough to get out of the obstacle's way and it subsequently crashes into the obstacle just before $11.5$ s.

[2]Previously these problems where solved up to thirty times slower than real-time [10, 16].

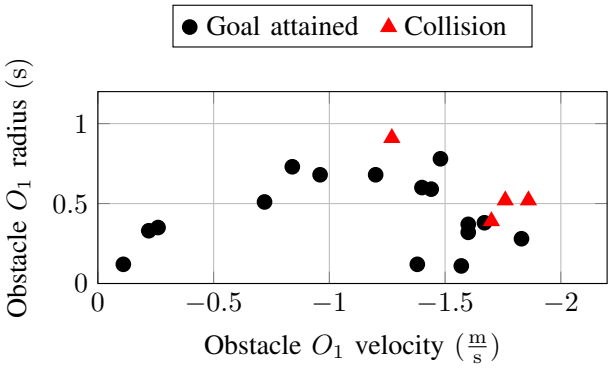



Fig. 5: The effect of obstacle size and speed.

### B. Person Following in an Unknown Environment

Fourteen experiments are run in an unknown environment to test the ability of the complete system shown in Figure 2. Table II shows the meta-data that was collected from these fourteen tests. These experiments are conducted on uneven grass near obstacles and over ditches and curbs, and on asphalt over train-tracks and near buildings and cars.

The remainder of this section provides detailed results of two of the fourteen tests cases. These results are shown in Figure 7b and Figure 7c. So, for simplicity, the obstacles in Figure 7b and Figure 7c are omitted, except for the obstacle on top of the person that is being followed.

Throughout each of these tests, the perception algorithm described in Section II-A provides accurate obstacle and person data to the trajectory planning algorithm. Given this data, trajectory planning problems are solved to generate the actual vehicle trajectories for speed and steering (see Figure

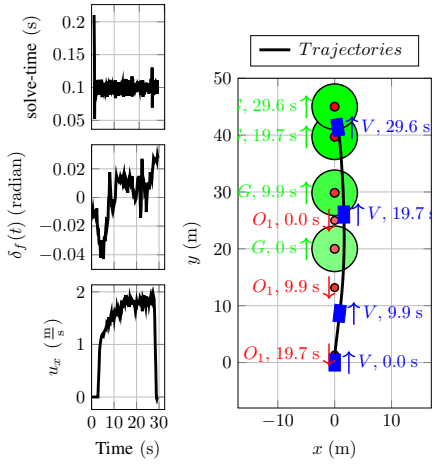
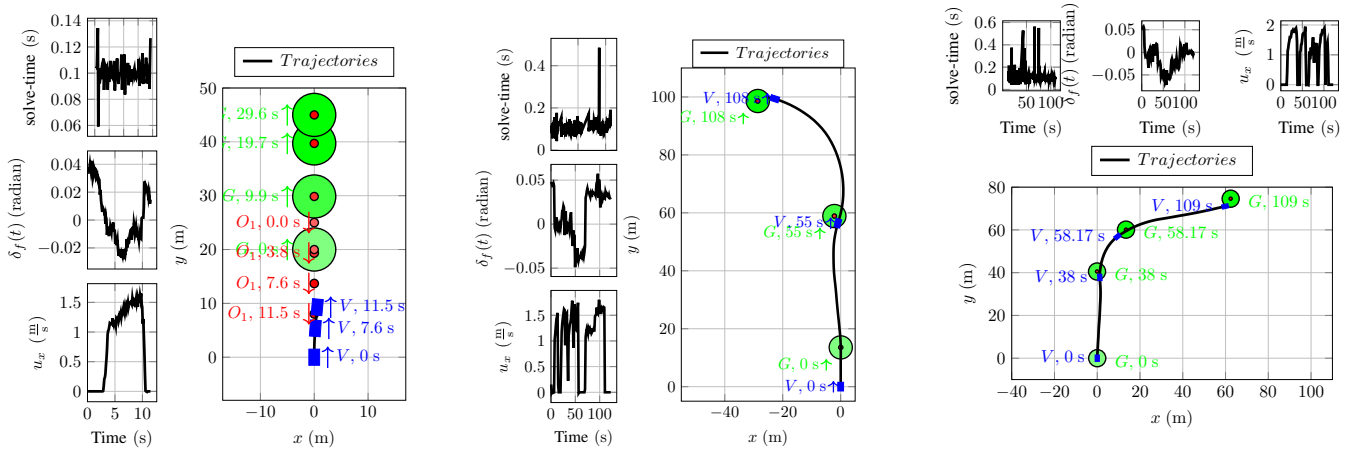



Fig. 6: A case with a larger obstacle traveling at moderate speeds, where the goal is attained. Blue indicates the autonomous vehicle, red indicates an obstacle, green indicates the goal region.




(a) (b) (c)

Fig. 7: (a) A case with a medium-sized obstacle traveling at moderately-high speeds, where the obstacle crashes into the vehicle. (b) A case where the person starts in front of the vehicle and walks forward across a ditch, then stops at 55 s and then turns left and walks across another ditch and then up a hill. (c) A case where the person starts beside the vehicle and walks across an uneven grassy field that eventually slops downwards to the right at roughly 15 degrees.

TABLE II: Meta-Data Collected in an Unknown Environment

| Parameter | Average Value |
|---|---|
| Distance | 60.7 m |
| Total Time | 69.5 s |
| Speed | 1.41 $\frac{\text{m}}{\text{s}}$ |
| Acceleration | 0.0884 $\frac{\text{m}}{\text{s}^2}$ |
| solve-time | 0.123 s |

7b and Figure 7c). Finally, despite the additional challenge of solving these trajectory planning problems in an unknown environment [23], the average solve-times for each case are about 0.106 s. This shows that our algorithm can handle a considerable increase in the number of obstacles[3] without having a large impact on computation time.

## VI. Conclusion

This paper introduces an NMPC-based trajectory planning algorithm for a person following AATV. This algorithm (1) simultaneously optimizes speed and steering, (2) minimizes control effort, and (3) avoids collisions with both static and dynamic obstacles. Experimental tests evaluate safety in both known and unknown environments. Results from these experiments show that

- the algorithm is solved in real-time (i.e., around 10 Hz) in both known and unknown environments,
- the algorithm helps the vehicle avoid obstacles that have a radius of less than 0.8 m traveling at speeds less than 1.5 $\frac{\text{m}}{\text{s}}$ in a known, dynamic environment,
- the algorithm is robust to disturbances such as ditches and curbs and
- the perception algorithm enables the planning algorithm to safely follow a person in an unknown environment.

Therefore, the proposed trajectory planning algorithm is found to be suitable for controlling a medium-sized, front-wheel-steered vehicle to safely follow a person in off-road, unknown, dynamic environments.

[3]It was common to observe approximately 20 obstacles at a give time.

## References


[1] A. I. Tarmizi, A. Z. Shukor, N. M. M. Sobran, and M. H. Jamaluddin, "Latest trend in person following robot control algorithm: A review," *Journal of Telecommunication, Electronic and Computer Engineering (JTEC)*, vol. 9, no. 3, pp. 169–174, 2017.

[2] J. J. Park and B. Kuipers, "Autonomous person pacing and following with model predictive equilibrium point control," in *2013 IEEE International Conference on Robotics and Automation*. IEEE, 2013, pp. 1060–1067.

[3] A. Leigh, J. Pineau, N. Olmedo, and H. Zhang, "Person tracking and following with 2d laser scanners," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 726–733.

[4] F. Vasconcelos and N. Vasconcelos, "Person-following uavs," in *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2016, pp. 1–9.

[5] D. Wooden, M. Malchano, K. Blankespoor, A. Howardy, A. A. Rizzi, and M. Raibert, "Autonomous navigation for bigdog," in *2010 IEEE International Conference on Robotics and Automation*. IEEE, 2010, pp. 4736–4741.

[6] G. Huskić, S. Buck, L. A. I. González, and A. Zell, "Outdoor person following at higher speeds using a skid-steered mobile robot," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 3433–3438.

[7] B. X. Chen, R. Sahdev, and J. K. Tsotsos, "Integrating stereo vision with a cnn tracker for a person-following robot," in *International Conference on Computer Vision Systems*. Springer, 2017, pp. 300–313.

[8] A. Cosgun, D. A. Florencio, and H. I. Christensen, "Autonomous person following for telepresence robots," in *2013 IEEE International Conference on Robotics and Automation*. IEEE, 2013, pp. 4335–4342.

[9] P. Nikdel, R. Shrestha, and R. Vaughan, "The hands-free push-cart: Autonomous following in front by predicting user trajectory around obstacles," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 1–7.

[10] J. Liu, P. Jayakumar, J. L. Stein, and T. Ersal, "Combined speed and steering control in high-speed autonomous ground vehicles for obstacle avoidance using model predictive control," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 10, pp. 8746–8763, 2017.

[11] H. Febbo, P. Jayakumar, J. L. Stein, and T. Ersal, "Real-time trajectory planning for automated vehicle safety and performance in dynamic environments," *arXiv preprint arXiv:2001.10163*, 2020.

[12] J. Liu, P. Jayakumar, J. L. Stein, and T. Ersal, "A study on model fidelity for model predictive control-based obstacle avoidance in high-speed autonomous ground vehicles," *Vehicle System Dynamics*, vol. 54, no. 11, pp. 1629–1650, 2016.

[13] S. Liu, N. Atanasov, K. Mohta, and V. Kumar, "Search-based motion planning for quadrotors using linear quadratic minimum time control," in *International Conference on Intelligent Robots and Systems*. IEEE, 2017, pp. 2872–2879.

[14] K. Mohta, K. Sun, S. Liu, M. Watterson, B. Pfrommer, J. Svacha, Y. Mulgaonkar, C. J. Taylor, and V. Kumar, "Experiments in fast, autonomous, gps-denied quadrotor flight," *arXiv preprint arXiv:1806.07053*, 2018.

[15] S. Liu, K. Mohta, N. Atanasov, and V. Kumar, "Search-based motion planning for aggressive flight in se (3)," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 2439–2446, 2018.

[16] H. Febbo, J. Liu, P. Jayakumar, J. L. Stein, and T. Ersal, "Moving obstacle avoidance for large, high-speed autonomous ground vehicles," in *American Control Conference*, 2017, pp. 5568–5573.

[17] C. Jewison, R. S. Erwin, and A. Saenz-Otero, "Model predictive control with ellipsoid obstacle constraints for spacecraft rendezvous," *IFAC-PapersOnLine*, vol. 48, no. 9, pp. 257–262, 2015.

[18] S. Ren, K. He, R. B. Girshick, and J. Sun, "Faster R-CNN: towards real-time object detection with region proposal networks," *CoRR*, vol. abs/1506.01497, 2015.

[19] J. Huang, M. Demir, T. Lian, and K. Fujimura, "An online multi-lidar dynamic occupancy mapping method," in *2019 IEEE Intelligent Vehicles Symposium (IV)*, June 2019, pp. 517–522.

[20] M. Kelly, "An introduction to trajectory optimization: How to do your own direct collocation," *SIAM Review*, vol. 59, no. 4, pp. 849–904, 2017.

[21] K. Kritayakirana and J. C. Gerdes, "Using the centre of percussion to design a steering controller for an autonomous race car," *Vehicle System Dynamics*, vol. 50, no. sup1, pp. 33–51, 2012.

[22] E. Velenis, P. Tsiotras, and J. Lu, "Aggressive maneuvers on loose surfaces: Data analysis and input parametrization," in *2007 Mediterranean Conference on Control & Automation*. IEEE, 2007, pp. 1–6.

[23] H. Febbo, "Real-time trajectory planning to enable safe and performant automated vehicles operating in unknown dynamic environments," Ph.D. dissertation, 2019.

[24] R. Rajamani, *Vehicle dynamics and control*. Springer Science and Business Media, 2011.

[25] J. Gonzales, F. Zhang, K. Li, and F. Borrelli, "Autonomous drifting with onboard sensors," in *Advanced Vehicle Control*. CRC Press, 2016, p. 133.

[26] J. Kong, M. Pfeiffer, G. Schildbach, and F. Borrelli, "Kinematic and dynamic vehicle models for autonomous driving control design," *IEEE Intelligent Vehicles Symposium, Proceedings*, vol. 2015-August, pp. 1094–1099, 2015.

[27] X. Zhang, A. Liniger, and F. Borrelli, "Optimization-based collision avoidance," *arXiv preprint arXiv:1711.03449*, 2017.

[28] G. Basset, Y. Xu, and O. Yakimenko, "Computing short-time aircraft maneuvers using direct methods," *Journal of Computer and Systems Sciences International*, vol. 49, no. 3, pp. 481–513, 2010.

[29] B. Ghannadi, N. Mehrabi, R. S. Razavian, and J. McPhee, "Nonlinear model predictive control of an upper extremity rehabilitation robot using a two-dimensional human-robot interaction model," in *International Conference on Intelligent Robots and Systems*. IEEE, 2017, pp. 502–507.

[30] J. Bezanson, S. Karpinski, V. B. Shah, and A. Edelman, "Julia: A fast dynamic language for technical computing," *arXiv preprint arXiv:1209.5145*, 2012.

[31] I. Dunning, J. Huchette, and M. Lubin, "Jump: A modeling language for mathematical optimization," *SIAM Review*, vol. 59, no. 2, pp. 295–320, 2017.

[32] J. Revels, "Reversediff," https://github.com/JuliaDiff/ReverseDiff.jl, 2017.

[33] A. Griewank, D. Juedes, and J. Utke, "Algorithm 755: Adol-c: a package for the automatic differentiation of algorithms written in c/c++," *ACM Transactions on Mathematical Software (TOMS)*, vol. 22, no. 2, pp. 131–167, 1996.

[34] H. Febbo, "NLOptControl," https://github.com/JuliaMPC/NLOptControl.jl, 2017.

[35] J. T. Betts, *Practical methods for optimal control and estimation using nonlinear programming*. Siam, 2010, vol. 19.