Kinodynamic Comfort Trajectory Planning for Car-like Robots

Heechan Shin[1] and Donghyuk Kim[1] and Sung-Eui Yoon[1]

***Abstract*— As personal autonomous mobility is getting to be more widely adopted, it is more important to consider comfortability of stuffs and persons carried by such mobility. In this work, we define the comfort of a trajectory as forces, specifically, translational force, received to objects carried by a robot while following the trajectory by measuring impulse. To maximize such a comfort, we propose a novel, kinodynamic comfort path planning method based on our definition of comfort. Our work is based on *direct collocation method* for handling our non-convex objective function. We also introduce Bidirectional Obstacle Detection(BOD) that identifies the distances along the perpendicular directions to the trajectory. This is mainly designed for avoiding obstacles while minimizing forces causing discomfort. Our experimental results show that our method can compute trajectories whose comfort measures can be up to 18 times higher than those computed by prior related objectives, e.g., squared velocity used for generating smooth trajectory.**

I. INTRODUCTION

Personal autonomous mobility or service robots are getting higher attention thanks to rapid advances on the related technology. While developing robotic hardware itself is important, considering objects and humans interacting with those robots are also important. Interaction with robots is getting more important, since various robots (e.g., Tesla self-driving cars) are readily available to us. Among many technical challenges, we focus on comfort of stuffs or humans carried by a robot during following the trajectory to the destination.

Generating comfortable trajectory for objects carried by a robot is related to controlling forces applied to them. In this regard, studying various dynamic properties and kinodynamic planning has been extensively studied [1]. Nonetheless, there have been relatively less work directly on defining and generating comfortable trajectories for a robot and its carried objects.

Main contributions. In this paper, we present a novel definition on the comfort and its counterpart concept, discomfort. Our discomfort metric directly measures forces, specifically, translational force, applied to the object carried by a robot (Sec. IV). Since our discomfort and other objective functions (e.g., obstacle avoidance) can be non-convex, we design our optimization framework based on the direct collocation approach, starting from a spline based initial trajectory using the interior-point method. We then use our novel obstacle avoidance method, Bidirectional Obstacle Detection (BOD), that reduces causing discomfort during our iterative optimization, while avoiding obstacles (Sec. IV-B).

[1]Heechan Shin (shin_heechan@kaist.ac.kr), Donghyuk Kim (donghyuk.kim@kaist.ac.kr), and Sung-eui Yoon (corresponding author, sungeui@gmail.com) are at School of Computing, Korea Advanced Institute of Science and Technology (KAIST)

To show the benefit of our approach experimentally, we have implemented our comfort kinodynamic planner and compared its performance against other approach. Thanks to the generality of our optimization framework, we were able to compare our objective function with other prior metrics. Our experimental result shows that our method has up to 18 times higher comfort values than those prior metrics. Furthermore, our method does not exceed an acceptable comfort limit while generating reasonably short travel time, while the variance of forces, which is intuitively related to the concept of comfort, received during following our trajectory is significantly lower, i.e., 1:6 to 1:90, than others (Sec. V).

II. RELATED WORK

In this section, we review previous approaches directly related to our work.

A. Path Planning for Comfortable Trajectory

Generating a comfortable trajectory is an important issue, especially when a robot delivers its fragile carried objects. To address this problem, a few works [2]–[4] have been studied to consider the comfort of a target, mainly about human.

Morales et al. [2], [3] proposed a human-comfort factor map, which represents human safety (e.g., distance to obstacles and visibility) and comfort factors according to linear velocity and acceleration of a wheelchair. In Gulati et al. [4] also dealt with comfort as their cost function by regarding a weighted sum of travel time and integration of jerk and angular derivatives as a comfort factor. The reason why they formulate such a cost function of the comfort is based on designing of road [5], railway vehicles [6] and movement of human arm [7]. Furthermore, they presented a formulation that can be used in a specific configuration, such as given start/goal velocity and acceleration.

Most previous works including aforementioned studies [2]–[4] and additionally [8] focused only on the comfort of human feeling. However, what we need to consider about is not only human comfort, but applied forces to anything that robots carry. In order to minimize forces that are applied to those target objects along a trajectory, we consider impulse (Sec. IV) during tracking the trajectory, resulting in improving comfort of those target objects.

B. Kinodynamic Planning

Our formulation of generating comfortable trajectory considers forces imposed to objects carried by a robot. As a result, considering dynamic properties of the robot is required when planning the trajectory, and thus our work is based on kinodynamic planning [1].

At a high level, there are mainly three orthogonal approaches to solve kinodynamic planning. The first approach is generating a smooth path using splines and then properly adjusting controls to follow the path. generates a smooth path that has continuous-curvature and then computes velocity/acceleration according to the generated path.

The second one is based sampling-based approaches, thanks to the success of various sampling methods (e.g., Rapidly-exploring Random Tree(RRT) [9]). One popular approach in this category is Kinodynamic RRT* [10]. This approach applies non-linear dynamics by linearizing the dynamics using the first-order Taylor approximation. On the other hand, Lee et al. [11] suggested a pre-computed database containing robot motions in accordance with dynamics and retrieve motions to extend an RRT-based random tree.

Optimization-based planning is the third category for solving the kinodynamic planning problem. Covariant Hamiltonian Optimization for Motion Planning (CHOMP) [12] is one of the most popular techniques in this category. Its cost function is a weighted sum of smoothness and obstacle avoidance, and is optimized by an iterative covariant gradient technique. On the other hand, Direct collocation method [13] transcribes the trajectory optimization problem into a non-linear program (NLP) and is widely used for the trajectory optimization [14], [15]. Trajectory replanning [14] uses the method when optimizing the trajectory represented by uniform B-splines.

Our work is based on trajectory optimization to handle the dynamic property of a robot, i.e., the direct collocation method that can optimize the trajectory efficiently. We give its background in Sec. III.

C. Obstacle Avoidance for Optimization-based Planning

Many optimization based planners [12], [14] use a distance field to avoid obstacles. Another way of avoiding obstacles is introduced by utilizing star-shaped obstacles [16]. In the latter case, trajectories are forced to be outside of the obstacle by making the distance from the trajectory to the center of the star-shaped obstacle to be larger than the distance from obstacle boundary to the center.

The aforementioned methods work well. However, for reducing discomfort further, we introduce a novel technique of avoiding obstacles named Bidirectional Obstacle Detection (BOD), which considers only perpendicularly local regions around the trajectory.

III. Background

Our work is based on the *direct collocation method* [13] for trajectory optimization. We briefly review its main concept in this section. Notations of terms are summarized in Table I and used throughout the paper.

A. Direct Collocation Method of Trajectory Optimization

Although some special cases of optimal control problems like Linear-Quadratic Regulator (LQR) has analytic solutions [17], generally optimal trajectories are generated by numerical methods because of the complexity of most applications [18]. At a high level, *Indirect method* and *direct method* are two main approaches of numerical methods for dealing with trajectory optimization problem. Among them, we discuss direct collocation method in this paper that our method is based on.

TABLE I
NOTATIONS

Notation	Description
N	Last index of zero-indexed collocation points
$x(t)$	State at time t
$u(t)$	Control at time t
$f(t,x(t),u(t))$	System dynamics, $\dot{x} = f(t,x(t),u(t))$
t_k	Time at k^{th} collocation point. Subscripted by k means at time t_k
t_f	Travel time
$\mathcal{C}$, $\mathcal{O}$	Comfort objective and obstacle objective
$h(t_0,x_0,u_0,t_N,x_N,u_N) = 0$	Boundary conditions of the trajectory

The main idea of the direct method is to convert the continuous trajectory optimization problem into a discrete non-linear program (NLP), which is called *transcription*. To this end, the trajectory is divided into several points named *collocation points*. The *direct collocation method* then interpolates those collocation points with splines, which are curves defined piecewise-polynomials.

In the transcription part, we consider three things. The first one is approximating an objective function, $J(\cdot)$. The objective function is usually composed of terminal and integral objectives. To approximate the integral objective, many integral approximation methods are available e.g., Simpson's rule. For easy implementation and computational speed, we use the basic trapezoidal quadrature method in our work (Sec. IV-C).

The second component of the transcription is about system dynamics. Dynamics of a robot are treated as constraints in the direct collocation method, and we thus rewrite the system dynamics into equality constraints of NLP. We convert the differential form of system dynamics to the integration form and then, similar to the approximation of integration above, approximate the system dynamics. Those approximated system dynamics between every pair of two collocation points are used as equality constraints of the NLP problem.

The last component is handling other constraints, e.g., boundary conditions, of the problem. This can be simply done by constraining all the collocation points instead of constraining functions of continuous time.

For the interpolating part, various interpolation methods can be applied. Since we use the trapezoidal collocation method in our work, controls and system dynamics are interpolated by linear approximation. The states are quadratically interpolated because states $x(t)$ are an integration of the system dynamics.

The main advantage of using the direct collocation method is that optimizing a vector of variables of NLP is easier than optimizing continuous functions of trajectory optimization problem [19] In addition, the resultant NLP of the tran-

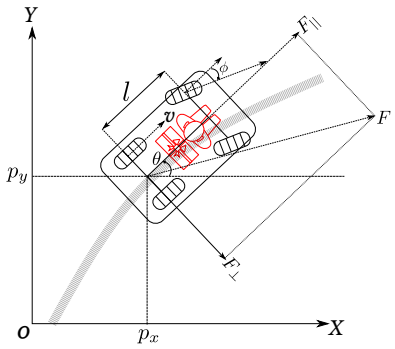


Fig. 1. A simple car-like model on the 2-D space. p_x and p_y are the reference positions of the car. θ and ϕ are angles of car body and steering. v is longitudinal velocity. l is a length between front and rear tire axises. $F_{||}$ and $F_\perp$ are longitudinal and centripetal forces, respectively, while F is a sum of them. Carried objects by the car are depicted as red; they can be anything like a person or a stuff.

scription is a *large-sparse* NLP in general [18], and thus many efficient large-scale-sparse NLP solvers [20]–[22] are available, thanks to its sparsity on the Jacobian and Hessian matrix of the objective function and constraints.

IV. Kinodynamic Comfort Planner

Our main goal of this work is generating a comfortable trajectory. Smooth paths have been widely studied [4], [8], [12], and thus can be candidates for such comfortable trajectory. Nonetheless, the smoothness does not necessarily mean the comfort, because a comfortable trajectory is a particular subset of smooth trajectories. As a result, for generating comfortable trajectory, we introduce a novel definition of comfort considering longitudinal and lateral force in Sec. IV-A. We then propose our obstacle avoidance method in Sec. IV-B, followed by our final transcribed objective at Sec. IV-C.

For the sake of simplicity, we explain our work on a simple, car-like model (Fig.1), but if the system dynamics are known, any robot models can be adopted to our work. The states and system dynamics of the car-like model are as follows:

$$x = \{p_x, p_y, v, \theta, \phi\}^T, \quad u = \{a, \omega\}^T$$

$$\dot{x} = f(t,x,u) = \left\{ v\cos\theta, v\sin\theta, a, \frac{v}{l}\tan\phi, \omega \right\}^T$$

m: mass of carried object

where p_x and p_y are positions in the 2D space and v is the tangential velocity. θ and ϕ are angles of the car body and steering, respectively (Fig. 1). a and ω are controls of the system, which are tangential acceleration and angular velocity of steering, respectively. All those values are functions of time, but we omit the parameter of time for simplicity, unless it is better to show the time parameter. Based on this model, we introduce an objective function of our planner that consists of travel time t_f, comfort objective $\mathcal{C}$, and obstacle objective $\mathcal{O}$ by defining a new definition of comfort and proposing a novel obstacle avoidance technique.

A. Definition of Comfort

The meaning of comfort in our work is not only limited to human feeling. Qualitatively speaking, we use the term of comfort to indicate *how low forces deforming states of a object carried by a robot are*, where the carried object could be human or stuffs, e.g., a person using personal mobility or food delivered by an autonomous vehicle which should be moved comfortably.

Intuitively speaking, the lower the forces are on the carried object, the more comfort the object feels. In this perspective, we also define its objective named *discomfort* that should be minimized to compute a comfortable trajectory. From now on, we focus on discussing how to minimize the discomfort, which is actually measured and used in our optimization framework.

Definition of Discomfort. Our definition of the discomfort is to measure the forces applied to the object. However, since the mass of the carried object does not vary during the travel, we are going to care only about accelerations from here. Because what we want to find is not exact 'comfortable value', but 'optimized trajectory', and we thus quantitatively measure *translational acceleration*. In the case of the simple car-like model (Fig. 1), $F_{||}$ is proportional to the longitudinal acceleration and $F_\perp$ is proportional to the lateral acceleration. If the mass, m, of the object is maintained along the trajectory, these accelerations are directly proportional to longitudinal and centripetal forces, respectively.

When a robot accelerates following the path, the carried object located in the robot receives a force in the opposite direction to the accelerations. Consequently, those opposite directional translational accelerations are proper to represent the discomfort of the carried object. Since the magnitude of the acceleration is mainly related to the discomfort, we finally define discomfort as the *squared magnitude of the translational acceleration*:

$$Discomfort = \tfrac{1}{m^2}\|F\|^2 = \tfrac{1}{m^2}\|F_{||} + F_\perp\|^2 = a^2 + \kappa^2 v^4,$$

where κ is the curvature of the trajectory and it can also be represented by $\frac{\tan\phi}{l}$.

Note that the curvature was also considered for prior methods generating smooth paths. For example, continuously changing curvature is essential to move smoothly especially when a car faces conjunction with a straight line and another curve; when the curvature of a trajectory is discontinuous, a robot has to stop wherever discontinuity occurs [23].

Our trajectory optimization method naturally maintains the continuous-curvature, because as we interpolate collocation points, curvatures associated with them are interpolated continuously thanks to the continuity of the tangent function between $\pm\frac{\pi}{2}$. On top of that, our definition takes a further step on measuring applied accelerations to the carried objects even on paths with continuous-curvatures.

Total Discomfort and Peak Discomfort. With the new definition of discomfort, there are two ways of measuring the discomfort of a trajectory. One is an integration of the discomfort along the trajectory, which is proportional to impulse, and the other one is measuring the peak value of the

discomfort. Since these two different ways are important, we consider both of the total and peak of discomfort of a trajectory within our optimization framework.

The total discomfort can be easily treated as an integration term of the objective function. On the other hand, the peak discomfort is not easy to deal with. This is mainly because just finding and reducing the maximum discomfort value of a trajectory may lead the NLP not to converge properly due to the discontinuity of derivatives of the max function [19]. Instead, with given minimizing the total discomfort, we treat the peak discomfort as a constraint:

$$\begin{aligned} &\text{minimize} \quad \int_0^{t_f} \mathcal{C}(t)\,dt = \int_0^{t_f} a^2 + \kappa^2 v^4\,dt \\ &\text{subject to} \quad a^2(t) + \kappa^2(t)v^4(t) < \mathcal{C}_{max}, \quad \forall t \in [0, t_f]. \end{aligned}$$

where $\mathcal{C}_{max}$ is a user-provided allowance on the peak discomfort.

Initial Trajectory for Interior-Point Method. To compute a trajectory satisfying our objective function, we use the Interior-Point Method (IPM), which is one of the popular non-linear optimization methods that can find a local optimum for non-convex problems [24].

Note that many motion planning problems belong to innate non-convex optimization category [12]. In these problems including ours, an initial guess on the trajectory is crucial not only for convergence, but also for where to converge. It is therefore desirable to start with a proper initial guess by taking account of our objective function.

We compute an initial trajectory in two steps. Firstly, we apply the cubic Hermite spline to generate a basic smooth path without considering any obstacles, for computing a smooth path with reduced curvature:

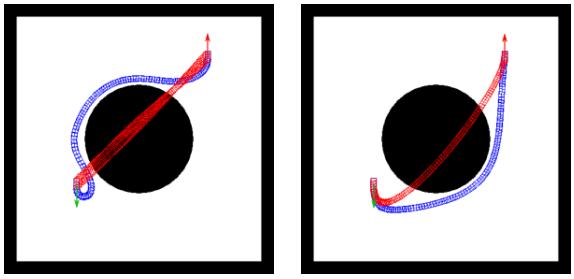
$$\text{Hermite spline } H(\hat{t}) = \begin{bmatrix} 1 \\ \hat{t} \\ \hat{t}^2 \\ \hat{t}^3 \end{bmatrix}^T \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -3 & -2 & 3 & -1 \\ 2 & 1 & -2 & 1 \end{bmatrix} \begin{bmatrix} p_0 \\ m_0 \\ p_1 \\ m_1 \end{bmatrix},$$

where, p_0 and m_0 are starting point and its tangent, and p_1 and m_1 are ending point and its tangent. $\hat{t} \in [0,1]$ is normalized parameter.

From the smooth spline, p_x, p_y and θ can be inversely calculated. The steering angle ϕ and longitudinal velocity v can be also calculated from the derivative of θ and (p_x, p_y), which are $\dot{\theta} = \frac{v}{l}\tan\phi$ and $(\dot{p_x} = v\cos\theta, \dot{p_y} = v\sin\theta)$, respectively.

Secondly, starting from the computed spline, we refine the trajectory by optimizing the objective function without considering obstacles using aforementioned optimization method, IPM. While obstacles are not considered, such initial trajectories can lead the final trajectory better for local planning.

Fig. 2 shows two different types of initial trajectories shown in red: linearly initialized trajectory (a) and proposed trajectory (b), given a circular obstacle. We refine those initial trajectory based on our kinodynamic comfort planner for computing our final trajectory shown in blue. The tested two different methods converge to different optima, due to the non-convexity of the configuration obstacle space, even though the obstacle is geometrically convex. Besides, our initial trajectory converges to a more comfort trajectory; the discomfort of a final trajectory starting from our initial trajectory is 65.75% less than the value of a trajectory staring from the linearly initialized trajectory in the above case.



(a) Linearly interpolated (b) Our spline based one

Fig. 2. Two comfortable trajectories against a simple circular obstacle. Red and blue trajectories are initial and final trajectories. (a): the states are linearly initialized. (b): the states are initialized based on our method.

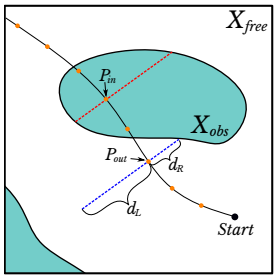


Fig. 3. Example of Bidirectional Obstacle Estimation. X_{free} is free space and X_{obs} is obstacle space. The red dotted line is boundary search line from inside of an obstacle and the blue dotted line is boundary search line from outside of an obstacle. Orange points are the collocation points. P_{in} is a collocation point which is inside of the obstacle. P_{out} is a collocation point which is outside of the obstacle.

B. Avoiding Obstacles with Minimum Discomfort

Starting from the initially created trajectory, we refine it, while considering obstacles. In many trajectory optimization methods, obstacle avoidance is achieved by iteratively pushing the trajectory away from obstacles. To perform the process while reducing generating any additional discomfort, we propose a novel way of avoiding obstacles, named Bidirectional Obstacle Detection (BOD), which pushes the trajectory perpendicularly to the trajectory on collocation points.

We optimize our trajectory by pushing collocation points of the trajectory perpendicular to the trajectory (Fig. 3). The reason why we use the perpendicular direction for pushing the points is to minimize an effect, e.g., changes of velocity and acceleration, of avoiding obstacles. In this regard, prior trajectory optimization methods, e.g., CHOMP [12], project their workspace gradient of distance function, which is obtained commonly by signed distance filed, at each collocation point orthogonally to the movement direction of the trajectory.

To realize our goal effectively, our BOD method uses a new distance function, $d(x)$, whose gradient is directly

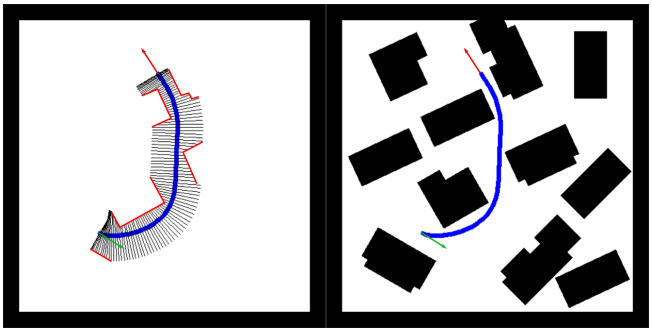


Fig. 4. Left: observed obstacle boundaries (red) by our BOD method. Right: the original global map and trajectory (blue).

perpendicular to the trajectory. When pushing collocation points of the trajectory perpendicular to the direction of the movement, which is same to the direction of the trajectory, the change of velocity, caused by obstacle avoidance, is minimized because inner product between the moving direction and pushing direction is almost zero which is similar to orthogonal force has no effect to the displacement. Minimizing the velocity change caused by avoiding obstacles is important to attain comfort.

Our BOD method uses a discretized map on the environment like occupancy maps [25] that can be constructed from sensor data. Our method aims to detect obstacle boundaries in two orthogonal directions at each collocation point perpendicular to the trajectory. To efficiently perform the obstacle detection, we uses the Bresenham's Algorithm [26], an well-known traversal method on regular structures.

For each collocation on the trajectory, the tangent vector is identical to its θ. Consequently, we can compute two perpendicular lines at the point of the trajectory in the 2D space. One is on the left-hand side $(\theta + \frac{\pi}{2})$ and the other one is right-hand side $(\theta - \frac{\pi}{2})$ of the trajectory. We call these perpendicular lines as *search vectors*, $\vec{s}$; the left and right search vectors are denoted by $\vec{s}^L$ and $\vec{s}^R$, respectively. Search vectors can be easily extended to a 3D workspace by generating a number of search vectors that are laid on a perpendicular disk to the trajectory. Nonetheless, we focus on handling the 2D simple car model in this paper.

We detect obstacles on the discretized map along the search vector starting from each collocation point using the Bresenham's algorithm. Like ray-tracing technique, $\vec{s}$ walks the map and stops when it meets an obstacle boundary. Also, we use a search threshold, ε_s, for terminating the map traversal, when the obstacle boundary is located too far away. In other words, if any obstacles are not detected within a ε_s, the map traversal and detection is stopped. The left image of Fig. 4 shows an example of the detected obstacle boundary by BOD, given the input, global map shown in the right image.

Objective Function of Obstacle Avoidance. The way of measuring the distance to the obstacles is one of key components for effectively performing obstacle avoidance. In our work, we suggest a new distance function that does not require any projection to to the perpendicular line.

Note that if we use the signed distance field used in prior works [12], [14], [27], it does not provide the perpendicular distance to the trajectory for each collocation point, losing the orthogonality for minimizing the discomfort. Quantitatively, using our BOD approach shows meaningful improvements, i.e., up to 19.64% in terms of the accumulated forces over the signed distance field in our tested cases.

Our BOD computes distances along two search vectors, $\vec{s}^L$ and $\vec{s}^R$, and these two distances are denoted as d_L and d_R for the left and right sides; see Fig. 3. Depending on a position of each collocation point, it can be inside or outside of the obstacle; e.g., P_{in} and P_{out} in Fig. 3 are inside and outside an obstacle, respectively.

Intuitively speaking, when the point is within the obstacle, the trajectory should be pushed toward the shorter distance between $d_L(x)$ and $d_R(x)$. On the other hand, when the point is outside of obstacles, the trajectory can be pushed towards the larger distance.

While the intuition is simple, the gradient direction can be discontinuity, especially when the shorter distance is exchanged, e.g., from the left side to the right side, resulting in inability to converge. Instead of taking this naïve approach, we propose a new distance function that can consider both distances d_L and d_R, while maintaining continuity:

$$d(x) = \begin{cases} (d_L(x)+\varepsilon_d)(d_R(x)+\varepsilon_d) & if\ x \in X_{obs} \\ -(d_L(x)-\varepsilon_d)(d_R(x)-\varepsilon_d) & if\ x \in X_{free} \end{cases},$$

where ε_d is an acceptable distance to the obstacles.

Suppose that o^L and o^R are detected collisions along $\vec{s}_L$ or $\vec{s}_R$, respectively. Then, the gradients of the distance function $d(x)$ are then computed as the following:

$$\begin{aligned} &If\ x \in X_{obs} \\ &\quad \nabla d(x) = \frac{\partial d(x)}{\partial x} = \frac{\partial (d_L(x)+\varepsilon_d)(d_R(x)+\varepsilon_d)}{\partial x}, \\ &If\ x \in X_{free} \\ &\quad \nabla d(x) = \frac{\partial d(x)}{\partial x} = -\frac{\partial (d_L(x)-\varepsilon_d)(d_R(x)-\varepsilon_d)}{\partial x} \quad \text{if } \exists o^L \wedge \exists o^R. \end{aligned}$$

When $\exists o^L \wedge \nexists o^R$, $d_L(x)$ keeps itself as a variable, yet d_R becomes a constant not being affected by changing of x, leading the term of $(d_R(x) - \varepsilon_d)$ to be constant; similar changes to other special cases like only $\exists o^R$.

Note that the proposed distance function forms a polynomial equation at each collocation points. As a result, it is continuous, facilitating the convergence within our opitmization framework.

Based on the distance function, we now need to use it for our optimization objective in addition to our discomfort function. Fortunately, obstacle avoidance with the distance function is well established and we adopt an obstacle objective function, $\mathcal{O}(x)$, similar to the one used in CHOMP [12]:

$$\mathcal{O}(x) = \begin{cases} d(x)^2 & if\ d(x) \geq 0, \\ 0 & otherwise. \end{cases}$$

$$\nabla \mathcal{O}(x) = \frac{\partial \mathcal{O}(x)}{\partial x} = \begin{cases} 2d(x)\frac{\partial d(x)}{\partial x} & if\ d(x) \geq 0, \\ 0 & otherwise. \end{cases}$$

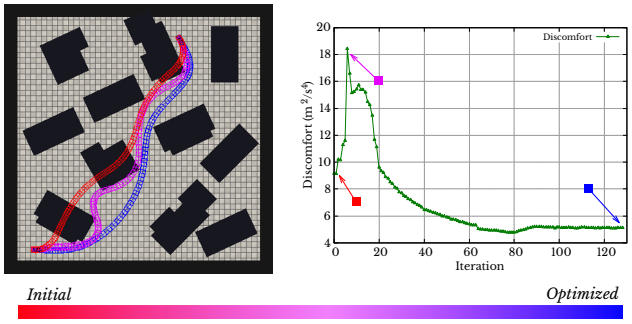


Fig. 5. The left image shows how the initial trajectory is refined as the number of iteration increases. The right graph shows the discomfort value as a function of the iteration.

Note that since our method directly considers the perpendicular distance to the trajectory, no projection procedure, performed in [12], is required. As a result, our method can achieve up to 45% less discomfort than the that computed by a signed distance field with the projection operation in practice.

C. Kinodynamic Comfort Planner

Summing up the aforementioned approaches and objective functions, we have the following, final transcribed NLP problem:

$$\underset{t_k, x_k, u_k \forall k}{\text{minimize}} \quad \lambda_{t_f} t_f + \sum_{k=0}^{N-1} \frac{(t_{k+1}-t_k)}{2}(\lambda_c(\mathcal{C}_k+\mathcal{C}_{k+1}) + \lambda_o(\mathcal{O}_k+\mathcal{O}_{k+1})),$$

subject to:

$$a_k^2 + \kappa_k^2 v_k^4 \ < \ \mathcal{C}_{max} \ \forall k,$$
$$h(t_0, x_0, u_0, t_N, x_N, u_N) = 0,$$
$$x_{k+1} - x_k - \frac{(t_{k+1}-t_k)}{2}(f_{k+1}+f_k) = 0, \ k = 0 \ldots N-1.$$

where λ_{t_f}, λ_c, and λ_o are weights of the travel time, our comfort objective, and obstacle avoidance objective, respectively. A terminal objective $\lambda_{t_f} t_f$ is added to consider the travel time, with other factors.

Fig. 5 shows how the discomfort value behaves as we have more iterations. The red trajectory of Fig. 5 shows the trajectory computed right after the first iteration. It has a low discomfort, but collides with the obstacles. Our planner pushes the trajectory to the magenta one outside of the obstacles using BOD at the expense of higher discomfort. Finally, it converges to the blue trajectory that has low discomfort without having any collisions.

V. Experiments

Our experiments are performed on an Intel i7 3.4GHz CPU with 16GB main memory. We use Interior Point OPTimizer (IPOPT v3.12.8) [28] package as our NLP solver. Although we mainly test the car-like robot (Sec. IV), many other mobile robots, e.g., omni-directional mobile robot or quadrotor, can be used thanks to the generality of our method.

Forces on a carried object are measured by the V-REP robot simulator [29] with the Bullet physics engine [30].

TABLE II

Experimental results; arc-length, Len., of the trajectory, accumulated and max forces (Σforce and Max f.).

			Scene1			
Objective	t_f	**ΣDiscomfort**	Len.	$\|v\|^2$	Σforce(σ^2)	Max. f.
$\mathcal{C}$(ours)	17.43	**5.12**	27.20	47.10	**8.70**(0.039)	**1.00**
$\mathcal{V} = \|v\|^2$	27.79	**11.84**	27.41	27.46	**7.46**(0.267)	**3.25**
$\mathcal{V}_f$ = F. $\|v\|^2$	17.43	**39.81**	29.80	51.80	**12.30**(1.300)	**7.36**
$\mathcal{T} = t_f$	15.68	**59.40**	30.29	59.29	**13.56**(2.219)	**8.52**
			Scene2			
Objective	t_f	**ΣDiscomfort**	Len.	$\|v\|^2$	Σforce(σ^2)	Max. f.
$\mathcal{C}$(ours)	12.31	**4.08**	14.41	18.60	**6.54**(0.046)	**0.98**
$\mathcal{V} = \|v\|^2$	14.68	**10.15**	14.32	14.16	**4.62**(0.534)	**4.33**
$\mathcal{V}_f$ = F. $\|v\|^2$	12.31	**14.37**	14.32	16.89	**5.50**(1.75)	**5.46**
$\mathcal{T} = t_f$	7.40	**73.76**	14.35	28.38	**8.49**(4.116)	**10.11**
			Scene3			
Objective	t_f	**ΣDiscomfort**	Len.	$\|v\|^2$	Σforce(σ^2)	Max. f.
$\mathcal{C}$(ours)	10.29	**3.30**	11.08	14.15	**5.33**(0.047)	**0.92**
$\mathcal{V} = \|v\|^2$	12.36	**13.31**	12.11	12.07	**7.63**(0.534)	**4.32**
$\mathcal{V}_f$ = F. $\|v\|^2$	10.29	**55.67**	13.48	19.44	**18.24**(1.743)	**5.76**
$\mathcal{T} = t_f$	6.64	**58.42**	12.18	22.97	**14.45**(3.01)	**8.50**

Experimental Setting. We use the same parameter values except unique parameters to each tested method for fair comparison. Static parameters are N = 100, convergence tolerance = 10^{-4}, resolution for grid map of BOD = 500 × 500, max iteration = 300, ε_s=1.2×robot width and λ_o = 100.

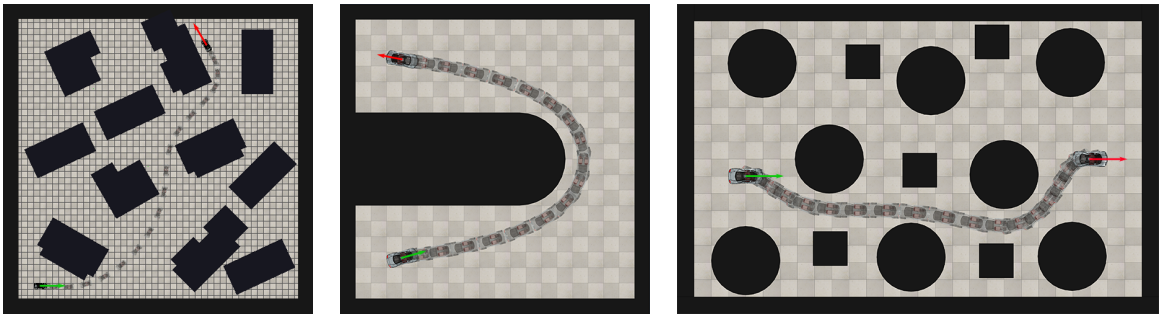
We set the start and goal velocity as zero, $v_0 = v_N = 0$ for our experiment, but these can be initialized to arbitrary numbers including negative ones indicating the backward motions. The mass of a carried object is set to 1 kg. The maximum comfort threshold $\mathcal{C}_{max}$ is dependent on the carried object. In our experiment, we assume it to be $(0.13g)^2$, which is approximately $1.63\text{m}^2/\text{s}^4$. According to the Hoberock's work [31], *"steady non-emergency accelerations in the range 0.11 g to 0.15 g fall in the 'acceptable' range for most studies."*, the comfort of passenger is set to squared $0.13g$; g is the gravitational acceleration.

We also apply BOD to all the tested methods for obstacle avoidance in our experiment and set parameters related to obstacle avoidance identically for fair comparison.

A. Comparisons with Other Objectives

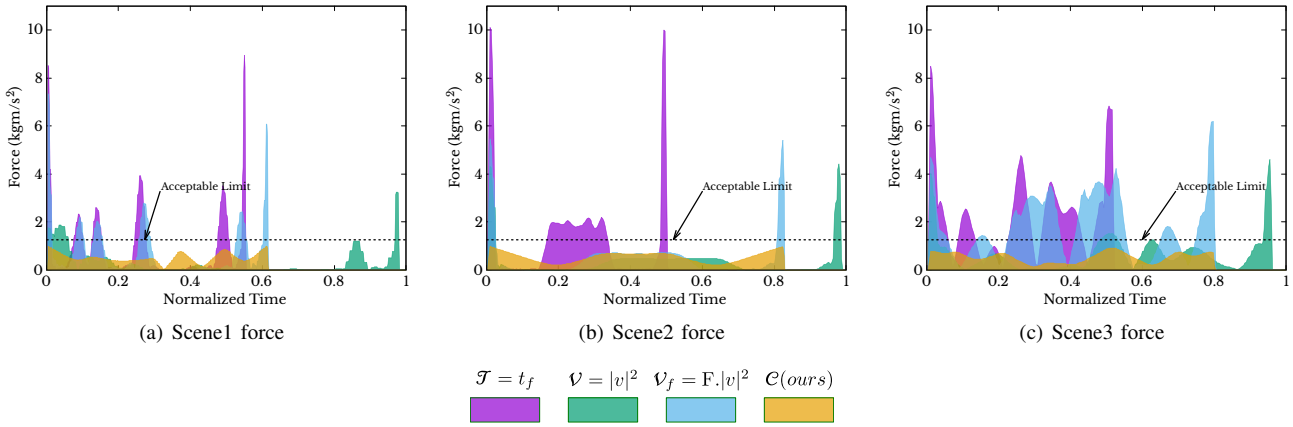
To demonstrate benefits and characteristics of our discomfort objective, we compare it with other widely used objectives. We use three different scenes shown in Fig. 6. Scene1 represents a large environment with scattered obstacles like buildings on downtown. Scene2 shows a cornering scenario where the comfort matters relatively more. The last scene is a cluttered environment like indoor office. Results on these scenes shows that our planner can be used as global comfort planner in various scenes, even though trajectory optimization is basically local planner. The final trajectory generated by our method is also depicted in Fig. 6.

To see their characteristics, we measure six different properties including our discomfort value, the squared velocity, $|v|^2$, commonly used in many prior trajectory optimization methods including CHOMP [12]. The reason why many prior



(a) Scene1 (b) Scene2 (c) Scene3

Fig. 6. Three test scenes and trajectories of our method. Green arrow is start direction and red arrow is goal direction. All the trajectories are generated with N=100, but for convenient to see, we depicted only 20 of them. (a) 25m×25m (b) 10m×10m (c) 15m×10m



(a) Scene1 force (b) Scene2 force (c) Scene3 force

Fig. 7. Force profiles of scenes normalized in a unit time interval. The maximum forces of our methods (orange) are always below the acceptable limit, $\approx 1.27\text{m/s}^2$, Sec. V-A. The maximum value of travel times (t_f) for each scene is (a) 28.5s (b) 15s (c) 13s.

methods consider $|v|^2$ is that when the travel time is fixed, minimizing $|v|^2$ flattens the path, making the path shorter and smoother. Statistics of the results are given in Table II. Other measures in the column header indicate total travel time (t_f) from the start state to the goal state, arc-length of the trajectory (Len), total summation of forces (Σforce) and maximum force (Max f.) received along the trajectory. Additionally, we report the variance of the forces measured at each collocation point over the trajectory in parenthesis next to the Σforce value.

We also test three other target objectives in addition to our discomfort objective, $\mathcal{C}$ (ours), within our optimization framework. Other tested objectives include the travel time, $\mathcal{T}$, and the squared velocity, $|v|^2$, denoted by $\mathcal{V}$. Minimizing $\mathcal{V} = |v|^2$ causes not only flattening the path but also increasing the travel time, because the velocity is proportional to the path length and inversely proportional to the time. Additionally, we consider $|v|^2$ with a constrained travel time, denoted as $\mathcal{V}_f$ = F. $|v|^2$. For $\mathcal{V}_f$, we fix the travel time with that of ours for the fair comparison with our method.

For our method, the weights for considering both the travel time and discomfort are set to 0.5. For the objective of $\mathcal{V}$, weights for the travel time and $|v|^2$ are also 0.5 and 0.5 for the fair testing; same to other objectives. For the objective $\mathcal{T}$, the weight for the travel time is 1.0. For all the different objectives, we use the same weight for the obstacle avoidance.

Table II shows experimental results with different objective functions across three tested scenes, and Fig. 7 shows profiles of forces according to time. There are two main observations that we would like to highlight. First of all, our method has the lowest discomfort across all the three scenes. The discomfort of our method is lower by up to 91%, 94%, 94% in each tested Scene 1 to 3 over using objective functions of $\mathcal{V}$, $\mathcal{V}_{fixed}$ and $\mathcal{T}$, respectively.

One may consider that having low discomfort values for our trajectories is a natural consequence, since our work mainly aims to minimize the discomfort value. To address this concern, we also measure the variance of forces received during following differet trajectory, as an intutive characteristic of discomfort of trajectories. While our method does not directly optimize against this measure, ours is significantly lower, 1:5.85 to 1:88.478, over those of trajectories computed by other objective functions.

Another interesting observation is about the forces. Our method shows reasonably low values of Σforce, and shows the lowest for Scene 3. For example, Σforce of our method in Scene1 is 16.6% higher than that of using the objective $\mathcal{V}$. However, the travel time of $\mathcal{V}$ takes one-half times more than that of ours. This indicates that using $|v|^2$ achieves the low force by moving the robot slowly. Fig. 7 shows force

profiles on trajectories.

In Scene 2, Σforce of our method is higher than $\mathcal{V}_{fixed}$, even if the travel time is same. One may conjecture that $\mathcal{V}_{fixed}$ generates a more comfortable trajectory than ours. However, note that the steady acceleration should be below $0.13\text{g}(\approx 1.27\text{m/s}^2)$ to be comfort, as mentioned earlier. While this constraint is satisfied by our method with $\mathcal{C}_{max}$, the max force of $\mathcal{V}_{fixed}$ is about two times higher than the acceptable acceleration threshold. Moreover, the max force of $\mathcal{T}$ is higher than $1\text{g}(\approx 9.8\text{m/s}^2)$. In other words, our method generates the most comfortable trajectories in the guideline of the Hoberock's work [31].

Travel time vs. Comfort. Depending on types of robots or carried objects, the importance of the travel time and discomfort can vary. Also, one can easily expect that as we reduce the travel time, we can get a more discomfort trajectory. In other words, depending on situations, we can utilize the trade-off between the travel time and discomfort within our optimization framework, because the weight of each component of our objective function is a user-definable.

VI. Conclusion

In this paper, we define a comfort objective and apply it to the trajectory optimization using direct collocation method for generating comfortable trajectory. We also propose a novel obstacle avoidance method called Bidirectional Obstacle Detection (BOD) which efficiently detects obstacles in the direction perpendicular to the trajectory. We have also observed that BOD successfully minimizes the effect on the trajectory i.e., change of velocity and acceleration, caused by obstacle avoidance during the optimization.

The experimental results show that the proposed method achieves not only the least discomfort but also the least maximum forces while tracking the generated trajectory. In some cases, the total forces applied to the object using other objectives outperform ours, however, they fail to minimize the travel time or received maximum force at the same time. Ours, however, is capable of achieving considerably low discomfort and travel time, which is more importance factor in practice.

In our future work, applying the state of the art NLP solver or considering dynamic environment can be included.

Acknowledgment

This research was supported by Next-Generation Information Computing Development Program(NRF-2017M3C4A7066317) and SW Starlab program(IITP-2015-0-00199).

References

[1] Bruce et al. Donald, "Kinodynamic motion planning", *Journal of the ACM (JACM)*, vol. 40, no. 5, 1993.

[2] Yoichi et al Morales, "Human-comfortable navigation for an autonomous robotic wheelchair", in *IROS*. IEEE, 2013.

[3] Yoichi et al Morales, "Including human factors for planning comfortable paths", in *ICRA*. IEEE, 2015.

[4] Shilpa et al Gulati, "A framework for planning comfortable and customizable motion of an assistive mobile robot", in *IROS*. IEEE, 2009.

[5] Jo Yung Wong, *Theory of ground vehicles*, John Wiley & Sons, 2008.

[6] Johan Förstberg, *Ride comfort and motion sickness in tilting trains*, PhD thesis, Institutionen för farkostteknik, 2000.

[7] Emanuel Todorov and Michael I Jordan, "Smoothness maximization along a predefined path accurately predicts the speed profiles of complex arm movements", *Journal of Neurophysiology*, vol. 80, no. 2, 1998.

[8] Paolo et al. Bevilacqua, "Path planning maximising human comfort for assistive robots", in *Control Applications (CCA), 2016 IEEE Conference on*. IEEE, 2016.

[9] Steven M LaValle, "Rapidly-exploring random trees: A new tool for path planning", 1998.

[10] Dustin J Webb and Jur van den Berg, "Kinodynamic rrt*: Asymptotically optimal motion planning for robots with linear dynamics", in *ICRA*. IEEE, 2013.

[11] Junghwan Lee, Heechan Shin, and Sung-eui Yoon, "Data-driven kinodynamic rrt", in *Advanced Robotics (ICAR), 2017 18th International Conference on*. IEEE, 2017, pp. 91–98.

[12] Matt et al. Zucker, "Chomp: Covariant hamiltonian optimization for motion planning", *The International Journal of Robotics Research*, vol. 32, no. 9-10, 2013.

[13] Nicola Bellomo, Bertrand Lods, Roberto Revelli, and Luca Ridolfi, *Generalized collocation methods: solutions to nonlinear problems*, Springer Science & Business Media, 2007.

[14] Vladyslav et al. Usenko, "Real-time trajectory replanning for mavs using uniform b-splines and a 3d circular buffer", in *IROS*. IEEE, 2017.

[15] Christoph Rösmann, Frank Hoffmann, and Torsten Bertram, "Kinodynamic trajectory optimization and control for car-like robots", in *IROS*. IEEE, 2017.

[16] Shilpa Gulati, *A Framework for Characterization and Planning of Safe, Comfortable, and Customizable Motion of Assistive Mobile Robots*, PhD thesis, University of Texas at Austin, 2011.

[17] Peter Dorato, Vito Cerone, and Chaouki Abdallah, *Linear quadratic control: an introduction*, Krieger Publishing Co., Inc., 2000.

[18] Harald Waschl, Ilya Kolmanovsky, Maarten Steinbuch, and Luigi Del Re, *Optimization and optimal control in automotive systems*, Springer, 2014.

[19] Matthew Kelly, "An introduction to trajectory optimization: How to do your own direct collocation", *SIAM Review*, vol. 59, no. 4, 2017.

[20] Philip E Gill, Walter Murray, and Michael A Saunders, "Snopt: An sqp algorithm for large-scale constrained optimization", *SIAM review*, vol. 47, no. 1, 2005.

[21] Christof Büskens and Dennis Wassel, "The esa nlp solver worhp", in *Modeling and optimization in space engineering*. Springer, 2012.

[22] Michael A Patterson and Anil V Rao, "Gpops-ii: A matlab software for solving multiple-phase optimal control problems using hp-adaptive gaussian quadrature collocation methods and sparse nonlinear programming", *ACM Transactions on Mathematical Software (TOMS)*, vol. 41, no. 1, 2014.

[23] Thierry Fraichard and Alexis Scheuer, "From reeds and shepp's to continuous-curvature paths", *IEEE Transactions on Robotics*, vol. 20, no. 6, 2004.

[24] Hande Y Benson, Robert J Vanderbei, and David F Shanno, "Interior-point methods for nonconvex nonlinear programming: Filter methods and merit functions", *Computational Optimization and Applications*, vol. 23, no. 2, 2002.

[25] Armin Hornung, Kai M. Wurm, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees", *Autonomous Robots*, 2013.

[26] Jack E Bresenham, "Algorithm for computer control of a digital plotter", *IBM Systems journal*, vol. 4, no. 1, 1965.

[27] Helen et al. Oleynikova, "Continuous-time trajectory optimization for online uav replanning", in *IROS*. IEEE, 2016.

[28] Andreas Wächter and Lorenz T Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming", *Mathematical programming*, vol. 106, no. 1, 2006.

[29] Eric Rohmer, Surya PN Singh, and Marc Freese, "V-rep: A versatile and scalable robot simulation framework", in *IROS*. IEEE, 2013.

[30] *Bullet Physics Library*.

[31] Lawrence L Hoberock, "A survey of longitudinal acceleration comfort studies in ground transportation vehicles", *Journal of Dynamic Systems, Measurement, and Control*, vol. 99, no. 2, 1977.