

Deformation Recovery Control and Post-Impact Trajectory Replanning for Collision-Resilient Mobile Robots

Zhouyu Lu, Zhichao Liu, and Konstantinos Karydis

Abstract—The paper focuses on collision-inclusive motion planning for impact-resilient mobile robots. We propose a new deformation recovery and replanning strategy to handle collisions that may occur at run-time. Contrary to collision avoidance methods that generate trajectories only in conservative local space or require collision checking that has high computational cost, our method directly generates (local) trajectories with imposing only waypoint constraints. If a collision occurs, our method then estimates the post-impact state and computes from there an intermediate waypoint to recover from the collision. To achieve so, we develop two novel components: 1) a deformation recovery controller that optimizes the robot's states during post-impact recovery phase, and 2) a post-impact trajectory replanner that adjusts the next waypoint with the information from the collision for the robot to pass through and generates a polynomial-based minimum effort trajectory. The proposed strategy is evaluated experimentally with an omni-directional impact-resilient wheeled robot. The robot is designed in house, and it can perceive collisions with the aid of Hall effect sensors embodied between the robot's main chassis and a surrounding deflection ring-like structure.

I. INTRODUCTION

Mobile robot motion planning algorithms can be classified in terms of the underlying optimization problem [1]. When there exist obstacles, most algorithms generate collision-free trajectories. Yet, it may be possible that potential collisions can in fact be useful [2]–[9], and thus an increasing number of research efforts aims at developing collision-inclusive motion planning algorithms [10]–[13]. This paper focuses on the latter category of collision-inclusive motion planning.

Development of collision-inclusive motion planners requires three core capabilities: 1) collision resilience, 2) collision identification, and 3) post-impact characterization. Advances in material science and design have helped introduce a range of collision-resilient mobile robots (e.g., [14]–[17]). For instance, passive protection devices can protect the robot from catastrophic impacts but cannot acquire information on where a collision occurred [7]; this is important information in order to predict how the robot will respond following a collision [5], [18]. Sensor-based collision detection and characterization methods have mostly focused on utilizing data from an onboard inertial measurement unit (IMU) [19]. However, IMUs are usually unable to distinguish collisions during aggressive maneuvers and to detect static contacts,

The authors are with the Dept. of Electrical and Computer Engineering, University of California, Riverside. Email: {zlu044, zliu157, karydis}@ucr.edu. We gratefully acknowledge the support of NSF #IIS-1910087, ONR #N00014-18-1-2252 and #N00014-19-1-2264, and ARL #W911NF-18-1-0266. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the funding agencies.

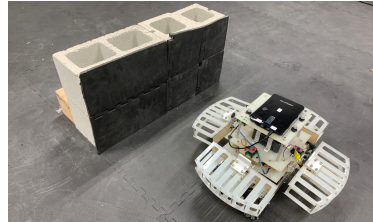


Fig. 1: Snapshot of our built-in-house omni-directional robot. The supplementary video contains detailed instances of our experiments.

resulting in low accuracy in collision detection. Hall effect sensors have been used in the past to provide more accurate collision detection based on estimated deformation [20]. In related yet distinct previous work [21], we have implemented a passive quadrotor arm design with Hall effect sensors, making the robot able to detect and characterize collisions. Herein, we embed a similar design to connect the main robot chassis and the deflection ring to estimate and characterize collisions. We design and deploy a distributed system of collision guards (that form a deflection ‘ring’ as shown in Fig. 1), where each surface is connected to the main robot chassis via passive visco-elastic prismatic joints.

Contrary to collision avoidance methods that generate trajectories only in conservative local space or require collision checking that has high computational cost and accurate detection of the obstacle, our method directly generates local trajectories based on a minimal effort optimization with only a waypoint constraint. Our method then estimates the post-impact state if a collision occurs and computes an intermediate trajectory to recover from the collision. The refined robot trajectory after collision attains a piece-wise polynomial form with respect to its flat outputs, and is computed online. Succinctly, in this work:

- We propose and solve a deformation control problem based on a dynamic model of wheeled robot visco-elastic collisions with polygon-shaped obstacles.
- We present a post-impact trajectory replanning algorithm based on constrained quadratic programming (QP) to locally adjust the trajectory after a collision.
- We develop and evaluate experimentally a deformation recovery and replanning (DRR) strategy to generate local trajectories based on any given waypoints.

II. RELATED WORKS

Most motion planning algorithms focus on generating collision-free trajectories. Collision-free algorithms handle obstacle avoidance in distinct ways. One way is to include geometric constraints of collision-free trajectories within the

optimization problem, when a map of the environment is exactly known [22]. When the map is partially-known, it is possible to formulate an obstacle-agnostic optimization problem [23], [24]. However, this approach is limited to short look-ahead planned trajectories, and may be unable to perform complex maneuvers around obstacles; when they do, this usually requires a computationally-expensive search to be able to generate a trajectory around obstacles. Another approach is to include the obstacles directly in the optimization problem, such as in the form of safe corridor constraints [25]. This way relies on decomposing the free (known or sensed) space as a series of P overlapping polyhedra. However, trajectories generated based on convex decomposition methods can be conservative since the solver can only choose to place the two extreme points of each interval in the overlapping area of two consecutive polyhedra. To overcome conservative solutions, it is possible to use binary variables to allow the solver to choose the specific interval allocation [26], but with increased computation time.

Importantly, as robots venture into more dynamic, irregularly-shaped and obstacle-cluttered environments, avoiding collisions becomes a major challenge [27], [28]. Employing a conservative local collision avoidance planner in cluttered environments may preclude the robot from finding a feasible path to the goal even if such path exists [29]. Further, detecting all obstacles in the environment can also be a challenging task, especially for those obstacles that are not opaque [9], such as glass or reflective surfaces. Developing collision-inclusive motion planners can help address the aforementioned challenges.

Post-impact characterization in collision-inclusive motion planning can be challenging to achieve, and it directly affects the form that post-impact (recovery) trajectories can attain. Estimating an accurate state of the robot after the collision is important for generating a reliable recovery trajectory. Yet, modeling collisions is a challenge and requires consideration of many interacting physical phenomena relating to the geometric, material, and inertial properties of each body involved in the collision; many of these properties are themselves difficult to model accurately. Related work [12] has introduced an empirical algebraic collision model, by directly relating pre- and post-impact velocities with no thrust commanded. This approach can only deal with a specific pair of objects over a relatively limited range of conditions. Other works redirecting the robot after a collision rely on the impulse contact model and feature different stages to track the recovery trajectory [30]. An impulse contact model assumes that velocity changes happen instantaneously, thus resulting in discontinuities in the recovery trajectory. The robot trajectory after contact with obstacles could be smoother if passive shock absorption devices were to be employed, e.g., as in [31], [32] albeit in a different context of robotic arms. Herein we develop a dynamic model of our wheeled robot after (visco-elastic) collisions based on the Voigt model for passive arms [31].

Compared to our previous work [11] that developed a higher-level waypoint planner that trades-off between risk

and collision exploitation, this paper proposes a new lower-level planner to generate local trajectories based on given waypoints and modify them accordingly if collisions occur.

Our proposed method shares similarities with blind navigation techniques (e.g., [20], [33]) in that it requires limited sensing capabilities (only mechanoeception via the passive compliant deflectors), and that it can be tuned to recover paths yielded by such methods as well. However, our proposed approach demonstrates features that distinct it from prior blind navigation methods; namely, its ability to exploit collisions in a controlled manner to improve overall collision-inclusive planning performance, its ability to be tuned to switch between exploration (i.e. explore free space after collision) and exploitation (i.e. follow obstacle surface after collision), and its compatibility to work alongside a range of other motion planners (it requires a list of waypoints that can be produced in any means).

III. THE DEFORMATION RECOVERY AND REPLANNING (DRR) STRATEGY

In contrast to collision avoidance algorithms, we do not impose any obstacle-related constraints in trajectory generation, nor we run a geometric collision check once a trajectory is generated. Instead, we directly generate a trajectory based on given waypoints.¹ If a collision occurs, the robot receives a signal that a collision has occurred from any of the Hall effect sensors embedded between the main chassis and its deflection surfaces and activates a collision recovery controller. The controller (described in Sec. IV) makes the robot detach from the collision surface and determines a post-collision state for the robot so as to facilitate post-impact trajectory replanning. The replanner (described in Sec. V) refines the initial trajectory since collisions change the continuity of derivatives of the trajectory followed before collision. To do so, the replanner uses the post-collision state determined by the recovery controller as the initial state for refined trajectory generation. The procedure repeats as new collisions may occur in the future, in a reactive and online manner. Our proposed *deformation recovery and replanning* (DRR) strategy is visualized in Fig. 2.

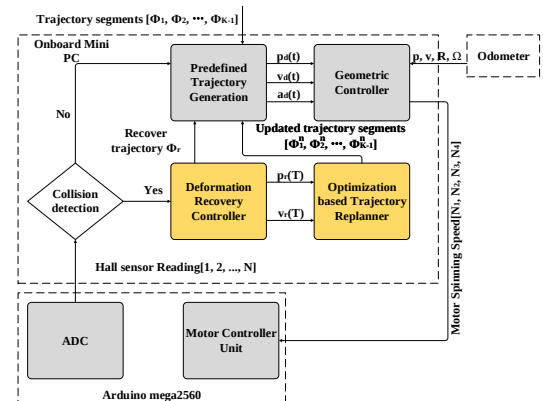


Fig. 2: Overview and software architecture of our DRR strategy for collision-inclusive motion planning and control. The method builds upon two novel components developed in this work; a deformation recovery controller and a post-impact trajectory replanner.

¹The list of waypoints can be computed via any path planning method.

IV. DEFORMATION RECOVERY CONTROL DESIGN

The purpose of our proposed deformation controller is to make the robot recover from a collision and reach a post-impact state that can facilitate recovery trajectory replanning (which we discuss in the next section). We first present important notation and working assumptions, and then move on to the development of the controller.

A. Problem Setting

Consider a holonomic mobile robot (Fig. 1), modeled as a point mass m . The robot's main chassis is connected to deflection surfaces via four passive visco-elastic prismatic joints (Fig. 3). Note that the springs inside each passive joint are pre-tensioned. The robot's compliant 'arms' can both protect the robot from catastrophic impact, and generate a external force driving it away from obstacles. External forces along each arm are caused via passive visco-elastic deformations assumed to follow the Voigt model; k and c denote the spring constant and damping coefficient, respectively. Hall effect sensors are used to measure the amount of deformation along each arm, and to signal collision detection when a user-tuned arm compression threshold is exceeded.²

We consider four key quantities related to spring lengths: neutral l_0 , pre-tensioned l_l , maximum-load l_e , and current l (also referred to as deformation vector). These quantities play a significant role in the deformation recovery controller; they are also summarized in Table I, along with other key notation. In single-arm collisions, current spring length vector l is aligned with the unit vector along the colliding arm, pointing from the tip of the arm to the center of robot. For clarity of presentation, we consider in the following single-arm collisions. In the case of multi-arm collisions, we compute individual contributions from each colliding arm's spring and then consider their vector sum as the compound deformation vector which is used in lieu of l .

We use three coordinate systems. The world and body frames (note ${}^w_b\mathbf{R}$ denotes the rotation matrix from body frame to world frame while ${}^b\mathbf{l}$ denotes the deformation vector expressed in the body frame), and a (local) collision frame $\mathcal{F}_c$. This frame is defined at the time instant a collision occurs, t_c , remains fixed for the duration of the collision recovery process, T , and then it is deleted. Its origin coincides with the origin of the robot when a collision is detected. Basis vector $\{\mathbf{n}, \mathbf{t}, \mathbf{k}\}$ of $\mathcal{F}_c$ are defined normal, tangent and upwards with respect to the deformation vector l . Let θ be the angle of deformation vector l in $\mathcal{F}_c$.

TABLE I: List of key notation used in the paper.

l_0	neutral length of the spring
l_s	pre-tensioned spring length (arm not compressed)
l_e	length at maximum spring load following Hooke's law.
l	current spring length (deformation vector)
k	spring constant of the arm.
c	damping coefficient of the arm.
${}^w_b\mathbf{R}$	Rotation matrix from body frame to world frame
${}^w_c\mathbf{R}$	Rotation matrix from collision frame $\mathcal{F}_c$ to world frame

²The threshold is tuned based on the sensitivity of the Hall effect sensors.

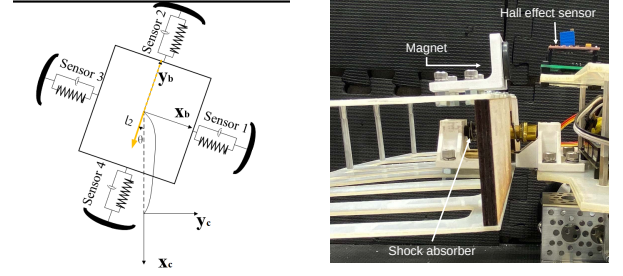


Fig. 3: (Left) Model of our wheeled robot equipped with compliant arms. (Right) Close-up view of the assembly of passive visco-elastic prismatic joint and Hall effect sensor.

We consider planar collision cases with polygon-shaped obstacles and assume that during deformation and until the collided arm recovers its initial length: 1) the tip of the arm remains in contact with the obstacle's collision surface but does not rotate about the z axis, and 2) the wheels of the robot contact the ground. The (frame-agnostic) robot collision dynamics is then given by

$$m\ddot{\mathbf{l}} + c\dot{\mathbf{l}} + k(\mathbf{l} - \mathbf{l}_0) = m\mathbf{a}_{in},$$

where $\mathbf{a}_{in}$ is the robot's body acceleration input.

B. Deformation Controller

The main idea underlying the proposed collision deformation recovery controller is to steer the post-impact state of the robot to a desired one within a time period of $[t_c, t_c + T]$. The time horizon T is an important hyper-parameter tuned by the user. Typically, longer T means the robot will recover from collision with longer time and smoother motion pattern; in this paper, we select $T = 0.5$ s.

We solve this problem by generating the state-space model of this problem first. Then we linearize the nonlinear state-space model using feedback linearization. We formulate a fixed-horizon (T) optimal control problem to solve for the control input of the linearized system. Finally, we discretize the linearized system fixed-horizon control problem and formulate it as a constrained quadratic programming problem.

The deformation controller operates with respect to the local, collision frame $\mathcal{F}_c$. Let the state variable be ${}^c\mathbf{x} = [x \ y \ \theta \ v_x \ v_y]^\top$. The control input is $\mathbf{u} = [u_x \ u_y \ u_\theta]^\top$, where $u_x = ({}^c\mathbf{a}_{in} - \frac{k}{m}({}^c\mathbf{l}_s - {}^c\mathbf{l}_0)) \cdot {}^c\mathbf{n}$, $u_y = ({}^c\mathbf{a}_{in} - \frac{k}{m}({}^c\mathbf{l}_s - {}^c\mathbf{l}_0)) \cdot {}^c\mathbf{t}$, and $u_\theta = {}^c\boldsymbol{\omega} \cdot {}^c\mathbf{k}$ with ${}^c\boldsymbol{\omega}$ being the angular velocity of the robot in the collision frame. Note that position control terms include compensation for the force caused by the spring being pre-tensioned when the robot's arm is at its rest length. Then, the state space model of the robot recovering from collision can be expressed as

$$\begin{cases} \dot{x} = v_x \\ \dot{y} = v_y \\ \dot{\theta} = u_\theta \\ \dot{v}_x = -\frac{k}{m}x - \frac{c}{m}v_x + u_x \\ \dot{v}_y = -\frac{k(\mu \text{sign}(v_y) + \tan \theta)x + f_0}{m} - \frac{c(\mu \text{sign}(v_y) + \tan \theta)v_x}{m} + u_y \end{cases} \quad (1)$$

where $f_0 = \mu k \text{sign}(v_y)({}^c\mathbf{l}_s - {}^c\mathbf{l}_0) \cdot {}^c\mathbf{n}$.

Since the robot is holonomic, we can decouple orientation from position control. In our approach we seek to make the robot keep the same orientation it has at the instant it collides throughout the collision recovery process. We follow this approach because it can simplify the overall deformation recovery control problem without sacrificing optimality.

The orientation and angular velocity errors during recovery time $t \in [t_c, t_c + T]$ are $e_R(t) = \frac{1}{2}(\mathbf{R}_d^\top \mathbf{R} - \mathbf{R}^\top \mathbf{R}_d)^\vee$ and $e_{\dot{R}}(t) = \boldsymbol{\omega} - \mathbf{R}^\top \mathbf{R}_d \boldsymbol{\omega}_d$, respectively.³ Index d denotes desired quantities; these are $\mathbf{R}_d = \mathbf{R}(t_c)$ and $\boldsymbol{\omega}_d = [0 \ 0 \ 0]^\top$. (All terms are with respect to collision frame $\mathcal{F}_c$.) Then,

$$u_\theta = -K_r e_{R,z}(t) - K_\omega e_{\dot{R},z}(t). \quad (2)$$

Note that since this is a planar collision problem, the collision recovery orientation controller considers only the z -components of orientation and angular velocity errors.

We now turn our attention to collision recovery position control. The translation-only motion in (1) is affine. Therefore, we can apply feedback linearization. The linearized system matrix $\mathbf{F}$ is

$$\mathbf{F} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -\frac{k}{m} & 0 & -\frac{c}{m} & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

with state vector $\mathbf{s} = [x \ y \ v_x \ v_y]^\top$. The control input matrix is $\mathbf{G} = \mathbf{I}_{2 \times 2}$ with control input vector $\boldsymbol{\nu} = [\nu_x \ \nu_y]$ given by

$$\begin{cases} \nu_x = u_x \\ \nu_y = u_y - \frac{k(\mu \text{sign}(v_y) + \tan \theta)x + f_0}{m} - \frac{c(\mu \text{sign}(v_y) + \tan \theta)v_x}{m} \end{cases} \quad (3)$$

We formulate an optimal control problem with fixed time horizon T based on the linearized system $\dot{\mathbf{s}} = \mathbf{F}\mathbf{s} + \mathbf{G}\boldsymbol{\nu}$. Using the change of variable $\tau = t - t_c$,⁴ we seek to solve

$$\min_{\mathbf{s}} \int_0^T (\mathbf{s}(\tau)^\top \mathbf{\Gamma} \mathbf{s}(\tau) + \boldsymbol{\nu}^\top(\tau) \mathbf{H} \boldsymbol{\nu}(\tau)) d\tau \quad (4a)$$

$$\text{subject to } \dot{\mathbf{s}} = \mathbf{F}\mathbf{s} + \mathbf{G}\boldsymbol{\nu}, \quad (4b)$$

$$- \|l_e - l_s\| \cos \theta \leq x \leq 0. \quad (4c)$$

$$\mathbf{s}(0) = [x_0 \ 0 \ v_{0,x} \ v_{0,y}]. \quad (4d)$$

$$\mathbf{s}(T) = [0 \ y_T \ v_{T,x} \ v_{T,y}]. \quad (4e)$$

Matrices $\mathbf{\Gamma} = \gamma \begin{bmatrix} \mathbf{I}_{2 \times 2} & \mathbf{0} \\ \mathbf{0} & \mathbf{0}_{2 \times 2} \end{bmatrix}$ and $\mathbf{H} = h \mathbf{I}_{2 \times 2}$ penalize the displacement during the recovery process and the control input, respectively. There is a trade-off between the displacement and the control input of the robot. Tuning parameters γ and h balance this trade-off to select the controller with minimal control energy and displacement.

Constraint (4c) dictates that the robot should be in contact with the collision surface until the colliding arm's spring

has recovered its original, pre-tensioned length l_s (i.e. the arm is no longer compressed) without compressing beyond its linear region l_e . Constraints (4d) and (4e) enforce initial and terminal position and velocity conditions, respectively. In detail, x_0 is determined by the colliding arm's Hall effector sensor reading. Since the vector form of the sensor's reading (that is, ${}^b \mathbf{l} - {}^b \mathbf{l}_s$) is expressed in the body frame, we need to transform it to the collision frame $\mathcal{F}_c$ as per

$$x_0 = -[1 \ 0] {}^w \mathbf{R}^\top {}^w \mathbf{R} ({}^b \mathbf{l} - {}^b \mathbf{l}_s). \quad (5)$$

The velocity components at the collision instant $v_{0,x}$ and $v_{0,y}$ are expressed in frame $\mathcal{F}_c$ and are estimated at run-time.⁵ Post-impact, the arm needs to be uncompressed (hence x_T is set to 0), but y_T is left as an unconstrained free variable. Post-impact terminal velocity components $v_{T,x}$ and $v_{T,y}$ are also expressed in $\mathcal{F}_c$ and can be set freely. In Alg. 1 lines 1–8, we discuss how to generate $v_{T,x}$ and $v_{T,y}$ based on the preplanned trajectory. We discretize the linearized system (4b) with sampling frequency $f = 10$ Hz using the Euler method, and solve the corresponding quadratic program with CVXOPT. The process is summarized in Alg. 1.

Computed control inputs (4) and (2) make the robot detach from the collision surface and help bring it to a temporary post-collision state which can be used as the initial condition for post-impact trajectory generation. We discuss this next.

Algorithm 1: Collision recovery control algorithm.

input : Displacement in body frame ${}^b \mathbf{l} - {}^b \mathbf{l}_s$ via Hall effect sensors readings; collision time instant $\tau_c \in [0, \Delta t_{ic}]$; position in world frame at collision instant, ${}^w \mathbf{p}_{\tau_c}$; velocity in world frame at collision instant, ${}^w \mathbf{v}_{\tau_c}$; rotation matrix ${}^w \mathbf{R}$; rotation matrix ${}^c \mathbf{R}$; next waypoint point in world frame, ${}^w \mathbf{p}_{next}$.

output : Control input $\mathbf{u}$

parameter: Maximum velocity of the robot v_{max}

- 1 ${}^w \mathbf{v}_T \leftarrow \frac{{}^w \mathbf{p}_{next} - {}^w \mathbf{p}_{\tau_c}}{\Delta t_{ic} - \tau_c}$
- 2 ${}^c \mathbf{v}_T \leftarrow {}^c \mathbf{R}^\top {}^w \mathbf{v}_T$
- 3 **if** ${}^c \mathbf{v}_{T,x} < 0$ **then**
- 4 | ${}^c \mathbf{v}_{T,x} \leftarrow 0$
- 5 **end**
- 6 **if** $\|{}^c \mathbf{v}_T\| \geq v_{max}$ **then**
- 7 | ${}^c \mathbf{v}_{T,y} \leftarrow v_{max} \text{normalize}({}^c \mathbf{v}_T)$
- 8 **end**
- 9 Calculate x_0 based on (5) with l_b
- 10 $v_{T,x} \leftarrow {}^c \mathbf{v}_{T,x}$, $v_{T,y} \leftarrow {}^c \mathbf{v}_{T,y}$
- 11 Calculate u_x , u_y based on (4) and (3) with given $v_{T,x}$, $v_{T,y}$ and x_0
- 12 Calculate u_θ based on (2)
- 13 **return** $\mathbf{u} \leftarrow [u_x \ u_y \ u_\theta]^\top$

³The vee map $\vee$ is the inverse of a skew-symmetric mapping.

⁴We employ this change of variable for clarity. Problem (4) resets every time a new collision occurs; this gives rise to an LTI system, hence the change of variable can apply.

⁵In the experiments conducted in this work, velocity measurements are provided via a motion capture camera system, but the method can apply as long as velocity estimates are available, e.g., via optical flow.

V. POST-IMPACT TRAJECTORY REPLANNING

We formulate the post-impact trajectory generation problem as a quadratic program with equality constraints. Letting $J' = \sum_{i=i_c}^I {}^w \mathbf{q}_{i,\beta}^\top {}^w Q_\beta^j(\Delta t_i) {}^w \mathbf{q}_{i,\beta}$, we seek to solve

$$\min_q \quad J(\mathbf{q}) = \sum_{i=i_c}^I \int_0^{\Delta t_i} \| {}^w P_{i,\beta}^j(t) \|^2 dt = J' \quad (6a)$$

$$\text{subject to} \quad {}^w \mathbf{A}_{0,i_c,\beta}^{(0)} \mathbf{q}_{i_c,\beta} = {}^w \mathbf{p}_{r,\beta}, \quad (6b)$$

$${}^w \mathbf{A}_{0,i_c,\beta}^{(1)} \mathbf{q}_{i_c,\beta} = {}^w \mathbf{v}_{r,\beta}, \quad (6c)$$

$${}^w \mathbf{A}_{\Delta t_I, I, \beta}^{(\alpha)} \mathbf{q}_{I,\beta} = {}^w \mathbf{d}_{\Delta t_I, I, \beta}^{(\alpha)}, \quad (6d)$$

$${}^w \mathbf{A}_{\Delta t_i, i, \beta}^0 \mathbf{q}_{i,\beta} = {}^w \mathbf{p}_{i+1,\beta}, \quad (6e)$$

$${}^w \mathbf{A}_{\Delta t_i, i+1, \beta}^{(\alpha)} \mathbf{q}_{i,\beta} = \mathbf{A}_{0, i+1, \beta}^{(\alpha)} \mathbf{q}_{i+1, \beta}. \quad (6f)$$

Superscript j denotes the derivative order; for example, $j = \{1, 2, 3, 4\}$ correspond to min-velocity, min-acceleration, min-jerk and min-snap trajectories, respectively. $\beta \in \{x, y\}$ indicates the x and y component of the trajectory. Δt_i is the time duration for i^{th} polynomial segment. Parameter $\mathbf{q}_{i,\beta}$ is the vector of coefficients of i^{th} polynomial. ${}^w \mathbf{A}_{0,i,\beta}^{(\alpha)}$ maps the coefficients to α^{th} order derivative of the start point in segment i , while ${}^w \mathbf{A}_{\Delta t_i, i, \beta}^{(\alpha)}$ maps the coefficients to α^{th} order derivative of the end point in segment i , for $\alpha \in \{0, 1, \dots, j-1\}$. Constraints (6b) and (6c) impose the initial values for the 0^{th} and the 1^{st} order derivatives to match the position and velocity values attained via the collision recovery controller, respectively. Constraint (6d) imposes that the α^{th} order derivatives of the end position are fixed. Constraint (6e) imposes that the trajectory will pass through desired waypoints after i_c . Constraint (6f) is imposed to ensure α^{th} continuity among polynomial segments.

We solve this QP problem given initial (post-collision) and end states, and intermediate waypoints. Then we perform time scaling as in [25] to reduce the maximum values for planned velocities and accelerations, as well as higher-order derivatives as appropriate, and thus improve dynamic feasibility of the refined post-impact trajectory.

A. Waypoint Adjustment

In some cases, we need to adjust the waypoints given in a preplanned trajectory with the information we get from the collision and then solve (6) with the adjusted waypoints. Such cases occur when there is no direct line of sight between the collision state and the waypoint at the end of the immediately next trajectory segment following collision recovery. By enabling such waypoint adjustment, the algorithm promotes exploration and in certain cases prevents the robot from being trapped in a local minima in which repeated collisions at the same (or very close-by) place could otherwise occur.

With reference to Alg. 2, we express in the local collision frame $\mathcal{F}_c$ the next waypoint ${}^w \mathbf{waypoint_list}[i_c + 1]$ (lines 1–3). In lines 4–8, we adjust the waypoint at the end of i_c segment if it lies on the same direction of ${}^c \mathbf{p}_r$ by moving it so to enable direct line of sight to collision state. In lines 9–17, if the waypoint at the end of i_c segment lies on the

opposite direction of ${}^c \mathbf{p}_r$, we insert a new intermediate waypoint in the waypoint list. The waypoint is generated by displacing ${}^c \mathbf{p}_r$ for a fixed (user-defined) ‘exploration distance’ $\epsilon_{explore}$ expressed in $\mathcal{F}_c$. In lines 18–23 we insert a new waypoint in the list as in lines 13–16 when there is direct line of sight with the waypoint at the end of i_c segment but the waypoint lies on the opposite direction of ${}^c \mathbf{p}_r$.

Algorithm 2: Waypoint adjustment algorithm

input : Displacement ${}^b \mathbf{l} - {}^b \mathbf{l}_s$; Position after the collision recovery in world frame, ${}^w \mathbf{p}_r$; waypoint list of preplanned trajectory; ${}^w \mathbf{waypoint_list}$; ${}^c \mathbf{R}$; trajectory segment i_c where the collision happens.

output : waypoint list after adjustment ${}^w \mathbf{waypoint_list}$

parameter: Robot radius ρ

```

1  ${}^w \mathbf{p}_{next} \leftarrow {}^w \mathbf{waypoint\_list}[i_c + 1]$ 
2 Transfer  ${}^w \mathbf{p}_{next}$  into  $\mathcal{F}_c$  frame to get  ${}^c \mathbf{p}_{next}$ 
3 Transfer  ${}^w \mathbf{p}_r$  into  $\mathcal{F}_c$  frame to get  ${}^c \mathbf{p}_r$ 
4 if  $-\rho \leq {}^c \mathbf{p}_{next,x} < {}^c \mathbf{p}_{r,x}$  then
5    ${}^c \mathbf{p}_{next,x} \leftarrow {}^c \mathbf{p}_{r,x}$ 
6   Transfer  ${}^c \mathbf{p}_{next}$  into world frame to get  ${}^w \mathbf{p}_{next}$ 
7    ${}^w \mathbf{waypoint\_list}[i_c + 1] \leftarrow {}^w \mathbf{p}_{next}$ 
8 end
9 if  $-2\rho \leq {}^c \mathbf{p}_{next,x} < -\rho$  then
10   ${}^c \mathbf{p}_{next,x} \leftarrow -2\rho$ 
11  Transfer  ${}^c \mathbf{p}_{next}$  into world frame to get  ${}^w \mathbf{p}_{next}$ 
12   ${}^w \mathbf{waypoint\_list}[i_c + 1] \leftarrow {}^w \mathbf{p}_{next}$ 
13   ${}^c \mathbf{p}_{add,x} \leftarrow {}^c \mathbf{p}_{r,x}$ 
14   ${}^c \mathbf{p}_{add,y} \leftarrow {}^c \mathbf{p}_{r,y} + \epsilon_{explore}$ 
15  Transfer  ${}^c \mathbf{p}_{add}$  into world frame to get  ${}^w \mathbf{p}_{add}$ 
16  Insert a waypoint  ${}^w \mathbf{p}_{add}$  between  $i_c$  and  $i_c + 1$ 
17 end
18 if  ${}^c \mathbf{p}_{next,x} < -2\rho$  then
19   ${}^c \mathbf{p}_{add,x} \leftarrow {}^c \mathbf{p}_{r,x}$ 
20   ${}^c \mathbf{p}_{add,y} \leftarrow {}^c \mathbf{p}_{r,y} + \epsilon_{explore}$ 
21  Transfer  ${}^c \mathbf{p}_{add}$  into world frame to get  ${}^w \mathbf{p}_{add}$ 
22  Insert a waypoint  ${}^w \mathbf{p}_{add}$  between  $i_c$  and  $i_c + 1$ 
23 end
24 return  ${}^w \mathbf{waypoint\_list}$ 
```

If a new waypoint is inserted in the list, we map the path generated by ${}^w \mathbf{p}_r$ and waypoints in the list after $i_c + 1$ into time domain using a trapezoidal velocity profile. If no new waypoint is inserted, we set the time duration of i_c segment in (6) as $\Delta t_{i_c} = t_{i_c+1} - t_c$, where t_{i_c+1} is the time reaching the next waypoint $\mathbf{p}_{i+1}$ per the preplanned trajectory.

VI. EXPERIMENTAL RESULTS

A. Robot and Environment Setup

We test our proposed algorithm with an omni-directional impact-resilient wheeled robot we built in-house (Fig. 1). The main chassis is connected to a deflection ‘ring’ via four arms that feature a passive visco-elastic prismatic joint each. Each passive arm has embedded Hall effect sensors to measure the length of the arm and detect collisions along each of their direction when the deformation exceeds a certain threshold.

Odometry feedback is provided by a 12-camera VICON motion capture system. The robot operates in an $2.0m \times 2.0m$ area with a rectangular pillar serving as a static polygon-shaped obstacles. An onboard Intel NUC mini PC (2.3 GHz i7 CPU; 16 GB RAM) processes odometry data and sends control commands to the robot at a frequency of 10 Hz.

The robot may flip when colliding with a velocity over an upper bound. To identify a theoretical collision velocity bound to avoid flipping, we use an energy conservation argument. Assume the kinetic energy before collision transfers into elastic potential energy of the passive arm, and the gravitational potential energy of the robot with small flipping angle counters the negative work input from the controller:

$$E_{k,t} - (v_{max}) = E_{ep}(l_e) - E_{ep}(l_s) + E_{gp}(\sigma_{max}) + m a_{in,max}(l_s - l_e).$$

Then,

$$v_{max} = \left\{ \frac{k[(l_e - l_0)^2 - (l_s - l_0)^2]}{2m} + g(\rho - l_s + l_e) \sin \sigma_{max} + a_{in,max}(l_s - l_e) \right\}^{\frac{1}{2}}.$$

The robot's radius is $\rho = 0.3$ m. The difference between the initial and neutral position of each passive arm is $l_s = 30.0$ mm; the maximum load length is $l_e = 15.0$ mm; and the neutral length is $l_0 = 41.5$ mm. The spring coefficient $k = 2.31$ N/mm. We select the largest flip angle $\sigma_{max} = 3^\circ$. The maximum acceleration input from the robot is $a_{in,max} = 5.0$ m/s². The mass of the robot is 6.0 kg. Then, we compute an upper theoretical velocity bound of $v_{max} \approx 0.7$ m/s.

The structure and number of passive arms affect collision detection. The most accurate detection occurs when the impact is located along the direction of arm. Collision detection accuracy can thus increase by adding more passive arms. Here, we use four arms to demonstrate how our proposed algorithm works, but future iterations will consider more arms, especially as we operate in more cluttered environments.

B. Experimental Testing of the Deformation Controller

To examine the deformation controller's effect in the overall trajectory generation method, we command the robot to collide with an obstacle and then apply the proposed deformation recovery controller. We perform 10 trials of each combination of input and output velocities (see Table II; variables with bars denote averages of observed values). We note that there are very few cases that collisions were not detected; only 9 out of 249.

Results suggest that the deformation controller generates a negative velocity to make the robot detach from the obstacle after collision. Actual output velocity $\bar{v}_{out}$ is determined by the actual input velocity $\bar{v}_{in}$ and the set output value $v_{out,set}$ though the latter may not be reached in practice. That is because feedback linearization is not robust to system parameter uncertainties that occur in practice. We observe that the velocity along $\mathbf{n}$ is closer to the set velocity than the velocity along $\mathbf{t}$. This is because most of the uncertainties in system parameters enter as unmodeled friction dynamics along the $\mathbf{t}$ axis. In future iterations we will upgrade the controller and the sensor setup to make the robot track the predefined velocity along $\mathbf{t}$ more accurately as well.

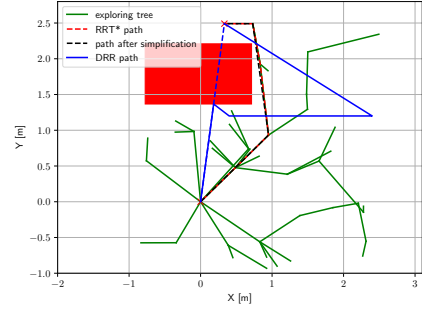


Fig. 4: An example of replanned path in case 2 using proposed DRR strategy and collision avoidance RRT* algorithm in [34].

However, it is sufficient for the recovery velocity to make the robot detach from the obstacle (which it does). Further, the sensor is more accurate when the input velocity is along $\mathbf{n}$; the average value of deformation detected is 29% larger.

C. Experimental Testing of the Overall DRR Strategy

We test our proposed DRR strategy with a trajectory generated based on solving an unconstrained QP problem and following time allocation and time scaling as in [25] without collision checking. We compare our algorithm's performance against the trajectory generation strategy in [34] with time allocation and scaling as in [25]. We compare the strategies under two cases: 1) when the previous path does not intersect with the collision surface; and 2) when the previous path intersects with the collision surface.

Case 1 tests the condition ${}^c p_{next,x} \geq -\rho$, i.e. no waypoint is added as per Alg. 2. Case 2 tests ${}^c p_{next,x} < -\rho$, i.e. a waypoint is added to the list. In case 2, we run RRT* to generate a collision free path and perform path simplification to remove nodes without affecting the path's collision safety (Fig. 4). The path simplification technique removes intermediate waypoints between two waypoints if a line segment between those two does not intersect with the obstacle. Then use the trajectory generation strategy in [34]. We perform 10 trials for each case. Instances of DRR and all experimental trajectories are shown in Fig. 5 and Fig. 6.

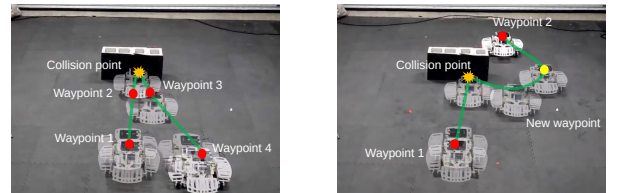


Fig. 5: Composite images of a sample experiment with our proposed DRR strategy. The robot must go from start to goal passing through all waypoints, including intermediate ones created post-collision. Snapshots shown every 2 s.

Even though we design a collision-free desired trajectory with the strategy in [34], the robot may still collide with the environment given for instance unmodeled dynamics such as drift. In case 1 there are 3 out of 10 trials that the robot in fact collides with the obstacle applying trajectory generation [34] that aims to avoid collisions. Table III shows statistics on mean arrival times, path lengths and control energy.

TABLE II: Deformation controller testing results.

$\mathbf{v}_{in,set}$	$\mathbf{v}_{out,set}$	$\bar{\mathbf{v}}_{in}$	$STD(\mathbf{v}_{in})$	$\bar{\mathbf{v}}_{out}$	$STD(\mathbf{v}_{out})$
(0.0, 0.5)	(0.0, -0.5)	(0.017, 0.532)	(0.013, 0.028)	(0.005, -0.372)	(0.019, 0.189)
(0.0, 0.7)	(0.0, -0.5)	(0.040, 0.709)	(0.015, 0.055)	(0.005, -0.387)	(0.021, 0.223)
(0.0, 0.5)	(0.0, -0.7)	(0.010, 0.500)	(0.012, 0.027)	(0.005, -0.340)	(0.030, 0.155)
(0.0, 0.7)	(0.0, -0.7)	(0.036, 0.709)	(0.016, 0.055)	(-0.016, -0.371)	(0.029, 0.223)
(0.0, 0.5)	(0.345, -0.345)	(0.022, 0.509)	(0.012, 0.027)	(0.017, -0.329)	(0.037, 0.213)
(0.0, 0.7)	(0.345, -0.345)	(0.034, 0.691)	(0.020, 0.053)	(0.010, -0.327)	(0.029, 0.201)
(0.0, 0.5)	(0.495, -0.495)	(0.023, 0.511)	(0.022, 0.040)	(0.027, -0.259)	(0.066, 0.192)
(0.0, 0.7)	(0.495, -0.495)	(0.027, 0.733)	(0.011, 0.056)	(0.046, -0.293)	(0.082, 0.305)
(0.0, 0.5)	(0.5, 0.0)	(0.032, 0.496)	(0.020, 0.047)	(0.016, -0.272)	(0.035, 0.188)
(0.0, 0.7)	(0.5, 0.0)	(0.036, 0.697)	(0.011, 0.062)	(0.033, -0.334)	(0.073, 0.248)
(0.0, 0.5)	(0.7, 0.0)	(0.017, 0.517)	(0.019, 0.015)	(0.048, -0.300)	(0.057, 0.217)
(0.0, 0.7)	(0.7, 0.0)	(0.026, 0.722)	(0.009, 0.024)	(0.014, -0.261)	(0.045, 0.219)
(0.345, 0.345)	(0.0, -0.5)	(0.313, 0.380)	(0.023, 0.025)	(-0.007, -0.113)	(0.043, 0.080)
(0.495, 0.495)	(0.0, -0.5)	(0.403, 0.485)	(0.048, 0.038)	(0.002, -0.126)	(0.027, 0.086)
(0.345, 0.345)	(0.0, -0.7)	(0.311, 0.372)	(0.020, 0.041)	(-0.019, -0.114)	(0.046, 0.086)
(0.495, 0.495)	(0.0, -0.7)	(0.417, 0.497)	(0.047, 0.034)	(0.010, -0.107)	(0.045, 0.086)
(0.345, 0.345)	(0.345, -0.345)	(0.281, 0.351)	(0.022, 0.014)	(-0.006, -0.061)	(0.040, 0.055)
(0.495, 0.495)	(0.345, -0.345)	(0.346, 0.472)	(0.052, 0.033)	(0.002, -0.166)	(0.042, 0.077)
(0.345, 0.345)	(0.495, -0.495)	(0.293, 0.367)	(0.036, 0.029)	(-0.009, -0.099)	(0.047, 0.060)
(0.495, 0.495)	(0.495, -0.495)	(0.353, 0.462)	(0.034, 0.036)	(0.006, -0.125)	(0.045, 0.086)
(0.345, 0.345)	(0.5, 0.0)	(0.261, 0.345)	(0.023, 0.038)	(0.023, -0.095)	(0.056, 0.099)
(0.495, 0.495)	(0.5, 0.0)	(0.394, 0.492)	(0.079, 0.129)	(0.031, -0.140)	(0.027, 0.100)
(0.345, 0.345)	(0.7, 0.0)	(0.309, 0.390)	(0.127, 0.166)	(0.013, -0.035)	(0.012, 0.030)
(0.495, 0.495)	(0.7, 0.0)	(0.351, 0.471)	(0.025, 0.040)	(0.002, -0.17)	(0.038, 0.108)

TABLE III: Comparison of trajectory generation strategy in [34] (Collision-avoidance) and DRR (Collision-inclusive) strategies.

	Strategy in [34]		DRR (our method)	
	Case 1	Case 2	Case 1	Case 2
$\bar{T}_{end}$ [s]	7.7	8.71	6.16	9.17
$STD(T_{end})$	0	2.20	0.22	0.31
$\bar{s}$ [m]	3.153	3.448	2.977	4.38
$STD(s)$	0.117	0.495	0.150	0.451
$\bar{E}_c$ [m ² /s ³]	56.83	80.62	58.42	255.96
$STD(E_c)$	29.34	54.38	32.92	133.54

In case 1 for DRR, mean arrival times $\bar{T}_{end}$ and path lengths $\bar{s}$ decrease by 25% and 6%, while the control energy increases by 2.8% on average. However, the error in the end point increases by 25%. In case 2, mean arrival times and path lengths increase by 5.2% and 27%, and control energy increases by 258%. This is because the output velocity of DRR is not flat since the robot needs to decelerate and then accelerate during the boundary following process. Moreover, the path generated by the boundary following is not the shortest. However, since the path between the collision point and the new inserted waypoint is close to the obstacle

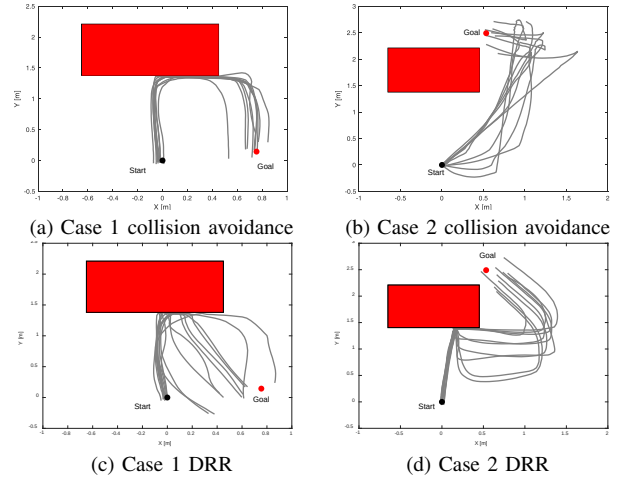


Fig. 6: Experimental trajectories generated from DRR and collision avoidance trajectory generation strategy in [34] when the preplanned path intersects or does not intersect with the obstacle. (In all cases we conduct 10 trials).

surface, the existence of the obstacle decreases the control error in free space. The error in the end point decreases by 12%. Overall, these results show the tradeoff between online reactive execution (whereby collision checking is skipped) and collision avoidance.

VII. CONCLUSIONS

The paper contributes to collision-inclusive motion planning, and proposes a new deformation recovery and re-planning strategy to generate local reactive replanned trajectories following a collision sensed by the robot at run-time. Two novel components make this strategy possible: 1) a deformation recovery controller that optimizes robot states during post-impact recovery, and 2) a post-impact trajectory replanner that adjusts the next waypoint with the information from the collision for the robot to pass through and generates a polynomial-based trajectory. Our proposed strategy runs online, and, given a sequence of waypoints that can be obtained in any manner, enables algorithmic collision resilience in a blind navigation paradigm.

Comparisons with a collision-avoidance trajectory generation method reveal fundamental tradeoffs between collision-avoidance and collision-inclusive motion planning and control. A key insight is that a collision-inclusive strategy may apply better when abrupt changes in robot motion are required (as in Case 1 testing), whereby collisions can be exploited to reduce mean arrival times and path lengths. We anticipate that exploiting collisions as described in this work can complement sensor-based autonomous navigation in cluttered environments where finding obstacle-free paths may be too computationally expensive.

REFERENCES

- [1] J. Tordesillas, B. T. Lopez, and J. P. How, "Faster: Fast and safe trajectory planner for flights in unknown environments," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2019, pp. 1934–1940.
- [2] T. Schmickl, R. Thoenius, C. Moeslinger, G. Radspieler, S. Kernbach, M. Szymanski, and K. Crailsheim, "Get in touch: cooperative decision making based on robot-to-robot collisions," *Autonomous Agents and Multi-Agent Systems*, vol. 18, no. 1, pp. 133–155, 2009.
- [3] K. Karydis, D. Zarrouk, I. Poulakakis, R. S. Fearing, and H. G. Tanner, "Planning with the star (s)," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2014, pp. 3033–3038.
- [4] D. W. Haldane, M. M. Plecnik, J. K. Yim, and R. S. Fearing, "Robotic vertical jumping agility via series-elastic power modulation," *Science Robotics*, vol. 1, no. 1, 2016.
- [5] Y. Mulgaonkar, A. Makineni, L. Guerrero-Bonilla, and V. Kumar, "Robust aerial robot swarms without collision avoidance," *IEEE Robotics and Automation Letters*, vol. 3, no. 1, pp. 596–603, 2017.
- [6] S. Mayya, P. Pierpaoli, G. Nair, and M. Egerstedt, "Localization in densely packed swarms using interrobot collisions as a sensing modality," *IEEE Trans. on Robotics*, vol. 35, no. 1, pp. 21–34, 2018.
- [7] A. Stager and H. G. Tanner, "Mathematical models for physical interactions of robots in planar environments," in *Intl. Symposium on Experimental Robotics*, 2018, in press.
- [8] N. Khedekar, F. Mascarich, C. Papachristos, T. Dang, and K. Alexis, "Contact-based navigation path planning for aerial robots," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2019, pp. 4161–4167.
- [9] Y. Mulgaonkar, W. Liu, D. Thakur, K. Daniilidis, C. Taylor, and V. Kumar, "The tiercel: A novel autonomous micro aerial vehicle that can map the environment by flying into obstacles," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2020, pp. 7448–7454.
- [10] Z. Lu and K. Karydis, "Optimal steering of stochastic mobile robots that undergo collisions with their environment," in *IEEE Intl. Conf. on Robotics and Biomimetics*, 2019, pp. 668–675.
- [11] Z. Lu, Z. Liu, G. J. Correa, and K. Karydis, "Motion planning for collision-resilient mobile robots in obstacle-cluttered unknown environments with risk reward trade-offs," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2020, pp. 7064–7070.
- [12] M. Mote, M. Egerstedt, E. Feron, A. Bylard, and M. Pavone, "Collision-inclusive trajectory optimization for free-flying spacecraft," *Journal of Guidance, Control, and Dynamics*, pp. 1–12, 2020.
- [13] J. Zha and M. W. Mueller, "Exploiting collisions for sampling-based multicopter motion planning," *arXiv preprint arXiv:2011.04091*, 2020.
- [14] A. Briod, P. Kornatowski, J.-C. Zufferey, and D. Floreano, "A collision-resilient flying robot," *Journal of Field Robotics*, vol. 31, no. 4, pp. 496–509, 2014.
- [15] D. W. Haldane, C. S. Casarez, J. T. Karras, J. Lee, C. Li, A. O. Pullin, E. W. Schaler, D. Yun, H. Ota, A. Javey *et al.*, "Integrated manufacture of exoskeletons and sensing structures for folded millirobots," *Journal of Mechanisms and Robotics*, vol. 7, no. 2, p. 021011, 2015.
- [16] A. Stager and H. G. Tanner, "Stochastic behavior of robots that navigate by interacting with their environment," in *IEEE Conf. on Decision and Control (CDC)*, 2016, pp. 6871–6876.
- [17] T. Li, Z. Zou, G. Mao, X. Yang, Y. Liang, C. Li, S. Qu, Z. Suo, and W. Yang, "Agile and resilient insect-scale robot," *Soft robotics*, vol. 6, no. 1, pp. 133–141, 2019.
- [18] A. Stager and H. G. Tanner, "Composition of local potential functions with reflection," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2019, pp. 5558–5564.
- [19] A. Battiston, I. Sharf, and M. Nahon, "Attitude estimation for collision recovery of a quadcopter unmanned aerial vehicle," *The Intl. Journal of Robotics Research*, vol. 38, no. 10-11, pp. 1286–1306, 2019.
- [20] A. Briod, P. Kornatowski, A. Klapotcz, A. Garnier, M. Pagnamenta, J.-C. Zufferey, and D. Floreano, "Contact-based navigation for an autonomous flying robot," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2013, pp. 3987–3992.
- [21] Z. Liu and K. Karydis, "Toward impact-resilient quadrotor design, collision characterization and recovery control to sustain flight after collisions," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2021.
- [22] J. A. Preiss, K. Hausman, G. S. Sukhatme, and S. Weiss, "Trajectory optimization for self-calibration and navigation," in *Robotics: Science and Systems*, 2017.
- [23] S. Liu, N. Atanasov, K. , and V. Kumar, "Search-based motion planning for quadrotors using linear quadratic minimum time control," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2017, pp. 2872–2879.
- [24] B. Zhou, F. Gao, J. Pan, and S. Shen, "Robust real-time uav replanning using guided gradient-based optimization and topological paths," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2020, pp. 1208–1214.
- [25] S. Liu, M. Watterson, K. Mohta, K. Sun, S. Bhattacharya, C. Taylor, and V. Kumar, "Planning dynamically feasible trajectories for quadrotors using safe flight corridors in 3-d complex environments," *IEEE Robotics and Automation Letters*, vol. 2, no. 3, pp. 1688–1695, 2017.
- [26] R. Deits and R. Tedrake, "Efficient mixed-integer planning for uavs in cluttered environments," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2015, pp. 42–49.
- [27] M. Hoy, A. S. Matveev, and A. V. Savkin, "Algorithms for collision-free navigation of mobile robots in complex cluttered environments: a survey," *Robotica*, vol. 33, no. 3, pp. 463–497, 2015.
- [28] S. Campbell, W. Naeem, and G. W. Irwin, "A review on improving the autonomy of unmanned surface vehicles through intelligent collision avoidance manoeuvres," *Annual Reviews in Control*, vol. 36, no. 2, pp. 267–283, 2012.
- [29] H. Oleynikova, Z. Taylor, R. Siegwart, and J. Nieto, "Safe local exploration for replanning in cluttered unknown environments for microaerial vehicles," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1474–1481, 2018.
- [30] G. Dicker, I. Sharf, and P. Rustagi, "Recovery control for quadrotor uav colliding with a pole," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2018, pp. 6247–6254.
- [31] T. Senoo, G. Jinnai, K. Murakami, and M. Ishikawa, "Deformation control of a multijoint manipulator based on maxwell and voigt models," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2016, pp. 2711–2716.
- [32] S. Tanaka, K. Koyama, T. Senoo, and M. Ishikawa, "Adaptive visual shock absorber with visual-based maxwell model using a magnetic gear," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2020, pp. 6163–6168.
- [33] N. Buniyamin, W. W. Ngah, N. Sariff, and Z. Mohamad, "A simple local path planning algorithm for autonomous mobile robots," *Intl. journal of systems applications, Engineering & development*, vol. 5, no. 2, pp. 151–159, 2011.
- [34] C. Richter, A. Bry, and N. Roy, "Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments," in *Robotics Research*, 2016, pp. 649–666.