Reshaping Viscoelastic-String Path-Planner (RVP)

Sarvesh Mayilvahanan
Department of Robotics
University of Michigan
Ann Arbor, MI, United States
smayil@umich.edu

Akshay Sarvesh
Electrical and Computer Engineering
Texas A&M University
College Station, TX, United States
sarvesh@tamu.edu

Swaminathan Gopalswamy
Mechanical Engineering
Texas A&M University
College Station, TX, United States
sgopalswamy@tamu.edu

***Abstract*—We present Reshaping Viscoelastic-String Path-Planner (RVP) a Path Planner that reshapes a desired Global Plan for a Robotic Vehicle based on sensor observations of the Environment. We model the path to be a viscoelastic string with shape preserving tendencies, approximated by a connected series of Springs, Masses, and Dampers. The resultant path is then reshaped according to the forces emanating from the obstacles until an equilibrium is reached. The reshaped path remains close in shape to the original path because of *Anchor Points* that connect to the discrete masses through springs. The final path is the resultant equilibrium configuration of the Spring-Mass-Damper network. Two key concepts enable RVP (i) *Virtual Obstacle Forces* that push the Spring-Mass-Damper system away from the original path and (ii) *Anchor points* in conjunction with the Spring-Mass-Damper network that attempts to retain the path shape. We demonstrate the results in simulation and compare it's performance with an existing Reshaping Local Planner that also takes a Global Plan and reshapes it according to sensor based observations of the environment.**

***Index Terms*—Path Planning, Motion Planning, Dynamic Control, Autonomous Navigation, Robotic Navigation**

I. Introduction

Autonomous Mobility of Robotic Vehicles (RVs) is a research problem that makes it's presence in a wide variety of applications. These applications include but are not limited to: Navigation applications like self-driving cars, Autonomous Trucking and Warehouse management. There is also a push to reduce the presence of humans in certain hazardous environments due to the dangers posed to their health and well-being. Some of these applications include using Robots to extinguish wildfires, Autonomous Mining, and rescue operations in disaster prone areas [1]. Oftentimes, due to the nature of the environment, lack of structure in the static and dynamic obstacles, the problem of *Autonomous Navigation of Robots* where a *RV* starts from some point in an environment and is given a mission to navigate to an end point remains a difficult one to solve. Some other applications of this are in the domain of Robot Manipulators whose applications vary widely from Robotic Surgery in the medical domain to folding clothes in the retail domain. These applications all require precise navigation of Robots in desired configurations while avoiding obstacles.

This work was supported in part by the Army Research Laboratories under Grant W911NF1920243.

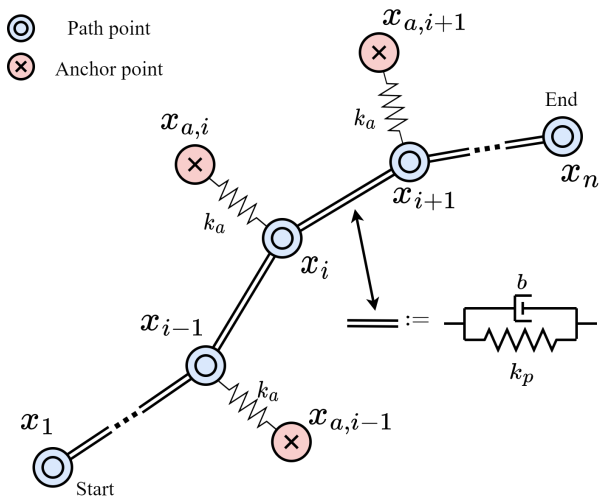


Fig. 1. *RVP system representation of path points (blue) and anchor points (red). The path points x_i are a discrete representation of the local path and are connected to adjacent path points via a spring and damper while the anchor points $x_{a,i}$ are connected to their corresponding path point via a spring.*

Autonomous Navigation of RVs can be classified into two over-arching categories: *Classical Techniques* and *Learning based techniques* as discussed in the survey paper [2].

Classical techniques rely on breaking down the problem of *Autonomous Navigation* into 2 main components: (i) *Perception* and (ii) *Navigation*. For perception, RV's use Proprioceptive and Exteroceptive sensors to get a view of the surrounding environment and know the where the RV is located within the environment. *Navigation* can further be split into 3 components: (a) *Path Planning*: Typically translates a higher level abstracted mission to a desired path that the RV needs to take. (b) *Motion Planning* translates this higher level plan into a series of *Motion Primitives* that the RV can then follow. (c) *Dynamic Control* translates the Motion primitives into low level actuator commands such as Acceleration, Steering wheel angle inputs, etc. *Classical techniques* usually rely on a model for the above components and perform a series of optimizations to achieve a goal of *Autonomous Navigation along a desired trajectory avoiding the obstacles*. This work would fall into the category of *Classical techniques*.

Learning Based techniques usually employ a data driven model using learning representations to come up with a policy

the can replace one of the components (or all of the components such as *Perception, Motion Planning, Path Planning and Dynamic Control*). The advantages of Learning based techniques include reduced modeling effort, which oftentimes come with a lack of explainability.

[3] introduced Probablistic Roadmap Method (PRM), a sampling based technique. [4] used the PRM technique to create an obstacle based PRM for 3D workspaces, which was one of the first of it's kind to implement the technique for motion planning. [5] introduced Rapidly Exploring Random Trees (RRT) which spawned variants such as RRT* [6].

Such abstractions can be used with graph based paradigms to obtain Coarse Global Planners like: Dijkstra's shortest path algorithm [7] and A* heuristic based search algorithm [8]. D* lite [9] is another optimized version of A* algorithm which avoids re-planning by using smart data structures to increase update speed and save computation cycles.

Local planners usually generate a sequence of finer way-points and/or low-level commands, which are fed directly to the controller to perform a smaller horizon based way-point following. As opposed to the global planner, the local planner needs to take into account the local features surrounding the robot like: obstacles, gradients, etc. The Local Planner is an Online Algorithm that needs to take quick decisions based on the immediate horizon.

Potential Field Methods (PFMs) were first introduced in [10]. Such an algorithm assumes a potential field that is being generated by the target points such that their gradients change towards the target results in an attractive force towards the target. Such a technique has potential of getting struck in Local Minima that could be generated by the potential fields and results in stagnation of the RV. [11] [12] [13] [14] [15] demonstrated optimizations on the existing algorithm to solve the issue of stagnations, but they mostly work under special conditions.

In the case of well-known vehicle dynamics and the observed environment, *Active Exploration Methods (AEMs)* such as Receding-Horizon Model Predictive Control (MPC) techniques have been used to solve online optimization problems for the purpose of navigation [16] [17] [18]. Such techniques can be used to generate low level control commands as well as a path for navigation. [16] used a MPC controller with a surround view camera to perform navigation and obstacle avoidance in a moving goal fashion. *Partially Observable Markov Decision Processes, (POMDP)* techniques can also be used [19] [20]. AEMs typically optimize an overall cost with respect to time horizons. AEMs are computationally expensive unless optimizations are done to avoid a large search space.

[21] introduced *Reshaping Local Path Planner*, that creates a Local Path based on a *Path Aware Moment Field*. This overcomes the disadvantages of the PFMs and avoids the issue of stagnation and is also computationally more efficient than AEMs by optimizing the action search space.

The classical techniques, albeit successful, require a large amount of engineering effort in modelling and designing the best Local Planner which works for every scenario. Oftentimes, the conventional techniques require extensive modeling which might be difficult to implement. To potentially reduce the amount of engineering effort, data intensive techniques using Machine Learning have been proposed in the literature. There are a large number of publications that address this problem in a few different ways. Based on the survey paper [2], learning-based navigation techniques can be classified into three parts: (i) Learning the Entire Navigation Stack, (ii) Learning Navigation Subsystems, (iii) Learning Individual Components. We list a few Learning based planners that are relevant to our research here [17] [22] [23] [24].

Functionally, Reshaping Viscoelastic-String Path-Planner (RVP) is similar to Elastic Band Planners (EBP) [25]. EBP updates and optimizes the Local path incrementally by following the Global path. The elastic band is deformed and it's shape and configuration is changed according to the internal contraction forces along the Global path and external repulsion from the obstacles around the path to remove any slack in the path. In this paper, RVP is compared with Reshaping Local Path Planner [21], which is also a Planner that reshapes a given Global Plan based on Sensor Observations of the Environment.

While other methods such as EBP employ the use of virtual forces that expand and contract the Local path, Reshaping Viscoelastic-String Path-Planner adds several key elements to produce stable and desirable results. The use of dampers within the Spring-Mass-Damper network provides a more steady reshaping of the path in reaction to (i) *Virtual Obstacle Forces*. Additionally, (ii) *Anchor points* within the system help to retain the shape of the desired Global path and provide key support for path curvature.

II. Preliminary Definitions

1) Operating Region Ω : Consider the operating region for the algorithm as being represented by a two-dimensional region Ω, and having properties defined on it, such as height, surface friction, etc.

2) Desired Global Path $\mathbf{x}^{dGP}$: We assume that an upstream *Global Path Planner* has determined a *Desired Global Path* given as a sequence of n_{GP} points $\mathbf{x}^{dGP} = \{x_i\}, x_i \in \Omega$, $i = \{1, ..., n_{GP}\}$. It is assumed that such a Desired Global Path is generated based on static or stale information. A distance parameterization of the desired global path will also be inferred from $\mathbf{x}^{dGP}$ by calculating the distance along the path: $\mathbf{s}^{dGP} := \{s_i^{dGP} = ||x_{i+1}^{dGP} - x_i^{dGP}||\} \forall i = \{1, ..., n_{GP} - 1\}, s_1^{dGP} = 0$.

3) Sensed Environment $\mathbf{x}^{\epsilon}$: Consider that all the sensors on-board the vehicle have been processed to generate a set of n_ϵ points $\mathbf{x}^{\epsilon} = \{x_i\}, x_i \in \Omega_\epsilon \subset \Omega, i = \{1, ..., n_\epsilon\}$. Associated with each point is a property function $\epsilon(x) : \Omega_\epsilon \rightarrow [0, c]$ for some positive constant c. In practice this property mapping could be from Ω_ϵ to integers, or just a boolean, in which case it defines a simple occupancy map. The sensed environment is generated at every control sample, and is considered fresh information. In this paper, the sensed environment is considered to already factor in uncertainties in the sensed information and is presumed to be the best estimate of the environment.

4) Local Path $\mathbf{x}^{lP}$: The output of the primary algorithms of the paper is a sequence of n_{lP} points $\mathbf{x}^{lP} = \{x_i\}, x_i \in \Omega$, $i = \{1, \ldots, n_{lP}\}$, such that there exists a corresponding smooth curve $\bar{\mathbf{x}}^{lP}(s) : [0, s_{lP}] \rightarrow \Omega$, and a monotonic sequence $\mathbf{s}^{lp} = \{s_i\}, s_i \in [0, s_{lP}]$ such that $\bar{\mathbf{x}}^{lP}(s_i) = \mathbf{x}^{lP}[i]$. Further, this curve shall be sufficiently smooth that a curvature can be defined at every point of the local path, i.e. there exists a well-defined curvature function $\bar{\kappa}^{lP} : [0, s_{lP}] \rightarrow \Re$, and from which we define the sequence κ^{lP} by $\bar{\kappa}^{lP}(s_i) = \kappa^{lP}[i]$. The path length is $p_{lP} := s_{n_{lP}}$

5) Anchor points $\mathbf{x}^a$: The formulation of the mass-spring-damper system involves a set of anchor points $\mathbf{x}^a = \{x_{a,i}\} \in \Omega$, i $\{1, \ldots, n_{lP} - 2\}$ corresponding to each of the local path points $x_i \in \mathbf{x}^{lP}$ excluding the start and end points. These anchor points maintain curvature in the path and tether the local path to the global path. Each anchor point $x_{a,i}$ spring has a spring length $l_{a,i}$.

6) Spring-Mass-Damper system: The viscoelastic string path $\mathbf{x}^{lP}$ consists of point masses with mass m. The spring and damping constants are defined based on two tuning constants ω and ζ: the spring constant between the path points $k_p := m\omega^2$, the spring constant for the anchor points $k_a := ck_p$, and the damping constant $b := 2m\zeta\omega$. Here, c is a scaling parameter representing the stiffness of k_a relative to k_p. The spring lengths $l_{j,k}$ are the intial spring lengths between two points $x_j, x_k \in \Omega$.

7) cross track error e^P : Given a path $\mathbf{x}^P$ and a point $y \in \Omega$, the cross track error $e^P(y)$ is the shortest distance from y to the path, obtained as $||y\vec{y}_\perp||$ where the point $y_\perp$ is a point on $\mathbf{x}^P$ such that the tangent to the path at that point is $\hat{t}_{y_\perp}$, and satisfies $y\vec{y}_\perp \perp \hat{t}_{y_\perp}$.

8) Path Deviation $\Delta P(\mathbf{x}^1, \mathbf{x}^2)$: Given two paths $\mathbf{x}^1$ and $\mathbf{x}^2$, the *Path Deviation* is defined as the cumulative sum of the square of the cross track errors: $\Delta P(\mathbf{x}^1, \mathbf{x}^2) := \sum_{x_i} e^2(x_i), \ \forall x_i \in \mathbf{x}^1$

9) Collision $\delta_c(x^P, x^\epsilon)$: Given a path $\mathbf{x}^P$ in a sensed environment $\mathbf{x}^\epsilon$, we define the boolean collision function δ_c as true if there is a collision between the path and any of the obstacles, else false. Specifically, we define $\delta_c = (||x_i^P, x_j^\epsilon||) > d_c, \ \forall x_i^P \in \mathbf{x}^P, x_j^\epsilon \in \mathbf{x}^\epsilon$, where d_c is a threshold distance used to determine if the path is too close to obstacles.

A. Problem Statement

Given a desired global path : $\mathbf{x}^{dGP}$, the problem objective is to synthesize a local path $\mathbf{x}^{lP}$ of length l_{lP} such that there are no collisions, i.e $!\delta_c$, and the Path Deviation $\Delta P(\mathbf{x}^{lP}, \mathbf{x}^{dGP})$ is minimized.

$$f_{\text{RVP}} : \mathbf{x}^{dGP} \times \Omega \rightarrow \mathbf{x}^{lp} \tag{1}$$

III. Reshaping Viscoelastic-String Planner

The Local path planner presented here employs the concept of an adaptive viscoelastic path that changes its shape based on applied virtual forces. These virtual forces are a function of the surrounding obstacles and the distance to those obstacles, which is represented by the sensed environment $\mathbf{x}^\epsilon$. For this implementation, the adaptive path is represented by a set of point masses connected in sequence by springs and dampers as shown in Figure 1. This serves to ensure the path points move as a collective, with adjacent points influencing the movement of one another. Additionally, the path points are anchored to the desired Global path by another set of springs to a set of Anchor points $\mathbf{x}^a$. These Anchor points tether the path points to the original path to ensure the resultant path is similar to the desired Global path and maintains the intended curvature.

The path points $x_i \in \mathbf{x}^{lP}$ are the set of points representing the path within the operating region Ω. The movement of these path points can be represented by the ordinary differential equation (ODE) (2).

$$m\frac{d^2x_i}{dt^2} + b\frac{dx_i}{dt} = F_{p,i} + \int_r \int_\theta F_{o,i}(r, \theta) \tag{2}$$

There are two virtual forces acting on the path points: a path-based force $F_{p,i}$ and an obstacle-based force $F_{o,i}$.

A. Forces

1) Path Force: The path-based force $F_{p,i}$ is the force acting on a point $x_i \in \mathbf{x}^{lP}$ by the adjacent points x_{i-1} and x_{i+1} through the connecting springs in addition to the force from the Anchor points by the point $x_{a,i} \in \mathbf{x}^a$. This force is calculated according to equation (3).

$$\begin{aligned} F_{p,i} = k_p(x_{i-1} - x_i - l_{i,i-1}) + k_p(x_{i+1} - x_i - l_{i,i+1}) \\ + k_a(x_{a,i} - x_i - l_{a,i}) \end{aligned} \tag{3}$$

Where k_p is the spring constant between the path points, k_a is the spring constant from the anchor points, and l is the spring length.

a) Anchor Point Locations: The Anchor point locations are calculated at the initial step of the simulation by solving for $x_{a,i} \in \mathbf{x}^a$, the location of the anchor point corresponding to the point $x_i \in \mathbf{x}^{lP}$, in equation (4). This point $x_{a,i}$ creates an equilibrium between the various spring forces. The Anchor points $\mathbf{x}^a$ serve to maintain the path's initial curvature and supply a force for the Local path to retain the shape of the desired Global path provides some incentive for the path to remain unchanged.

$$\begin{aligned} 0 = k_p(x_{i-1} - x_i - l_{i,i-1}) \\ + k_p(x_{i+1} - x_i - l_{i,i+1}) + k_a(x_{a,i} - x_i) \end{aligned} \tag{4}$$

$$l_{a,i} = d(x_{a,i}, x_i) \tag{5}$$

2) Obstacle Force: The obstacle force $F_{o,i}(r, \theta)$ is the force acting on the point x_i from an obstacle at distance r and angle θ. This force is calculated according to the piece-wise function (6).

$$F_{o,i}(r, \theta) = \begin{cases} a_2 \cdot \delta_{\mathbf{x}^\epsilon}(x_i, r, \theta) \cdot \frac{1}{r^n} & r \leq a_1 \\ a_2 \cdot \exp\left(\frac{-(r-a_1)}{a_3}\right) \cdot \delta_{\mathbf{x}^\epsilon}(x_i, r, \theta) \cdot \frac{1}{r^n} & r > a_1 \end{cases} \tag{6}$$

Where within a radius a_1, the obstacle force is as defined, and beyond which there is an exponential decay in the force affected by the tuning constant a_3. The obstacle force is inversely proportional to the distance r between the path point

and obstacle to the power n. The force is also scaled by the tuning constant a_2. Given a sensed environment $\mathbf{x}^\epsilon$, a boolean function $\delta_{\mathbf{x}^\epsilon}(x_i, r, \theta)$ is defined to be true if the point of distance r and orientation θ away from point x_i is occupied in the sensed environment $\mathbf{x}^\epsilon$ and false otherwise. This function ensures that the force only acts on the path point if there is an obstacle at a distance r and angle θ.

The profile of the obstacle force as a function of the distance r can be seen in Figure 2.

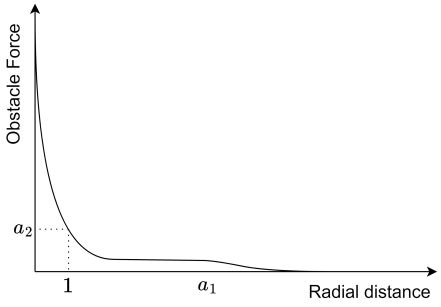


Fig. 2. *Obstacle force as a function of radial distance r for tuning parameters* a_1, a_2.

The force is very high at close distances to the obstacle and rapidly decreases due to the inverse proportionality to distance r. After the cutoff distance a_1, the force decays and eventually reaches a negligible magnitude. This ensures that obstacles further away from the path point enact little to no force.

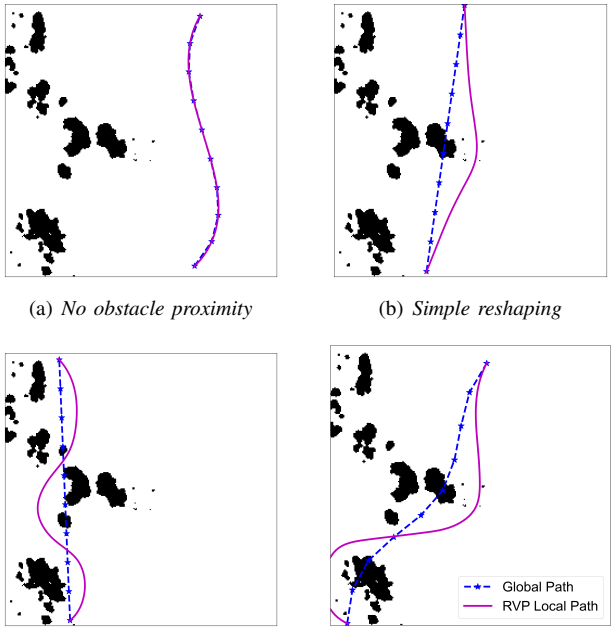
B. Resultant Path

RVP approximates the motion of the path points until an integration time t_f, which can also be represented by a maximum number of steps as in Algorithm 1. If the system detects that the path points $\mathbf{x}^{lP}$ have reached a steady-state prior to reaching t_f, it will automatically exit the simulation to decrease computational expense. The simulation considers the steady state to be satisfied if all path points have an acceleration below a certain threshold a_t: $\forall x \in \mathbf{x}^{lP}, \ddot{x} < a_t$.

After reaching the final state of the path, a spline-fit is performed on the path points and is interpolated to produce a set of evenly spaced points, which is the output of the Reshaping Viscoelastic-String Path-Planner .

The path planner is primarily evaluated based on its ability to provide a safe and stable path in a wide range of scenarios. In Figure 3, the Reshaping Viscoelastic-String Path-Planner is evaluated over four different global paths along with a *real Unstructured Sensed environment based on real Lidar data* obtained in an abandoned Quarry at the RELLIS campus of Texas A&M University. These select examples with varying difficulty highlight the large range of scenarios the planner can handle.

In Figure 3(a), the global path is not in close proximity to any obstacles. The resulting reshaped path from the planner therefore does not experience any forces from the obstacles and maintains the original path between due to the anchor points keeping the system in equilibrium. In Figure 3(b), there is one obstacle obstructing the global path, which causes the path to be pushed outward. However, the path forces ensure that the provided reshaped path does not deviate very far from the global path. In Figure 3(c), the global path must be reshaped significantly to pass through two key areas. The path points are able to reach a steady state passing in between the obstacles after an equilibrium is found between obstacle and path forces. Figure 3(d) indicates another example where the global path must undergo notable changes, in particular due to the oddly-shaped obstacle in the bottom left. The planner provides a path that avoids all obstacles and handles the complex obstacles.



(a) *No obstacle proximity* (b) *Simple reshaping*

(c) *Complex reshaping* (d) *Complex obstacle shape*

Fig. 3. *RVP performance over four scenarios over data from a Real-Unstructured Environment. The Global path is shown in dashed blue, the reshaped Local path is shown in solid magenta. (a) demonstrates the output of RVP when the desired global path is not in close proximity to any obstacles. (b) demonstrates the output of RVP when there is a single obstacle obstructing the desired global path. (c) & (d) demonstrate increasingly complex scenarios where multiple obstacles overlap the desired global path and there is a significant reshaping effort needed to avoid the obstacles.*

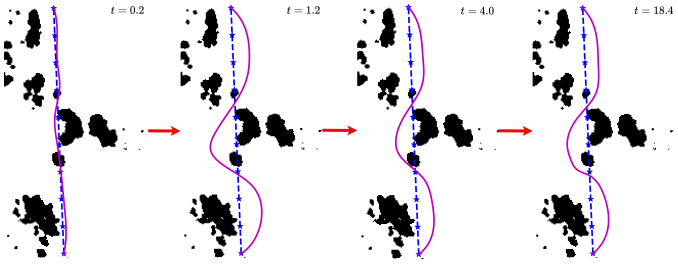


Fig. 4. *Visualization of dynamically reshaped Local path at various integration timesteps for scenario in Figure 3(c). The Global path is shown in dashed blue and the current state of the Local path in solid magenta.*

In Figure 4, the reshaped Local path is shown at select integration times $0 < t < t_f$ during the reshaping process. At

the start, the path points $x_i \in \mathbf{x}^{lP}$ are initially being pushed outward by the obstacle forces. In the second integration timestep, the path points reach their peak deviation after which the path forces pull the path points back towards the initial path in the third integration timestep. After some refinement, the Local path finally reaches a steady state unobstructed by obstacles due to the damping of the system. This dynamic movement of the Local path indicates how the presented path planner is robust to a variety of paths and sensed environments.

C. Planner Flow

Algorithm 1 provides a high level overview of the inner workings of this planner when finding a solution using discrete methods. In each step of simulating the path point movement, the path forces and obstacle forces are calculated and used in conjunction with dynamics to move the simulation forward.

Algorithm 1 Reshaping Viscoelastic-String Path-Planner

Data: $\mathbf{x}^{dGP}$: desired global path; $\mathbf{x}^{\epsilon}$: sensed environment; ω, ζ: system parameters; a_1, a_2: obstacle force parameters, p_{min}: minimum portion of steps to simulate
Result: $\mathbf{x}^{lP}$: local path
$\mathbf{x}^{lP} \leftarrow \mathbf{x}^{dGP}$
$k_p, k_a, b \leftarrow f(\omega, \zeta)$
Initialize anchor points, spring lengths
step $\leftarrow 1$
while *step* $<$ *max_steps* **do**
  $F_p \leftarrow$ Path forces $\forall x_i \in \mathbf{x}^{lP}$
  $F_o \leftarrow$ Obstacle forces $\forall x_i \in \mathbf{x}^{lP}$
  $\mathbf{x}^{lP} \leftarrow$ step_dynamics$(\mathbf{x}^{lP}, F_p, F_o)$
  if *path_stagnated*$(\mathbf{x}^{lP})$ **then**
    perturb_path$(\mathbf{x}^{lP})$ ▷ Perturb stagnated path point
  end
  if *step* $> p_{min} \cdot$ *max_steps and is_steady*$(\mathbf{x}^{lP})$ **then**
    return $\mathbf{x}^{lP}$ ▷ Path has reached steady state
  end
end
return $\mathbf{x}^{lP}$

The planner also addresses common problems encountered by similar Potential Field Methods such as getting stuck in Local Minima, which could result in the path getting stuck in undesirable locations. The functions *path_stagnated* and *perturb_path* identify path points that are stuck in local minima inside of obstacles (indicating a collision) and perturb them. This perturbation, in addition to the path forces from adjacent points, is able to pull the stagnated points into more favorable positions.

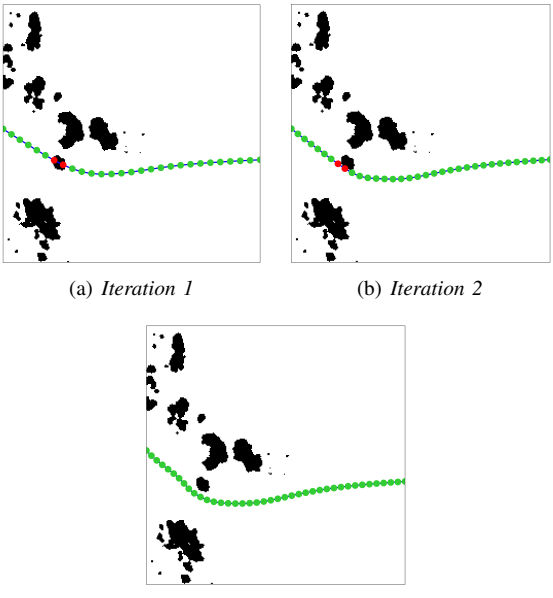
Additionally, the system simulates a minimum number of steps based on a user-defined variable p_{min}. If the system indicates that the path has reached a stable, steady-state solution after those minimum number of steps, no further simulation is needed and the path is returned. This is done in order to save computation time in the case of relatively simple reshaping.

D. Iterative Simulation

The path does not always reach a safe state after a single iteration of the simulation, so multiple iterations may be required. After each iteration, the path is evaluated for collisions. If the evaluation deems the path to be safe, the new path is returned. Otherwise, the unsafe path is used as the initial path for another iteration to obtain a safer path. Each successive iteration enforces a decay in the system parameters to ensure that later iterations do not drastically alter the changes from the previous iterations. Due to the decay in the algorithm's parameters, if more than a specified number of iterations are required to reach a safe path, the simulation is exited and the path is flagged as unsafe.

Algorithm 2 Iterative RVP

Data: $\mathbf{x}^{dGP}$: desired global path; $\mathbf{x}^{\epsilon}$: sensed environment; ω, ζ: system parameters; a_1, a_2: obstacle force parameters; λ: parameter decay; d_c: safety margin
Result: $\mathbf{x}^{lP}$: local path; flag: (1: safe, 0: unsafe)
$\mathbf{x}^{lP} \leftarrow \mathbf{x}^{dGP}$
iteration $\leftarrow 1$
while $\delta_c(\mathbf{x}^{lP}, \mathbf{x}^{\epsilon}, d_c)$ **do**
  $a_1 \leftarrow a_1 \cdot \lambda^{\text{iteration}-1}$
  $a_2 \leftarrow a_2 \cdot \lambda^{\text{iteration}-1}$
  $\mathbf{x}^{lP} \leftarrow$ RVP($\mathbf{x}^{lP}$, $\mathbf{x}^{\epsilon}$, ω, ζ, a_1, a_2)
  if *iteration* $=$ *max_iters* **then**
    return $\mathbf{x}^{lP}$, 0 ▷ Local path marked as unsafe
  end
  iteration $\leftarrow$ iteration $+ 1$
end
return $\mathbf{x}^{lP}$, 1 ▷ Local path is collision free



(a) *Iteration 1* (b) *Iteration 2*

(c) *Iteration 3*

Fig. 5. *Progression of model evaluation for a given path and* d_c. *The sensed environment* $\mathbf{x}^{\epsilon}$ *is represented by the obstacles in black. The evaluated points are marked green if deemed safe with no collisions* $(!\delta_c)$ *and marked red if unsafe* (δ_c).

1) Path Evaluation: The resultant path is evaluated at each iteration for collisions using the boolean collision function $\delta_c(\mathbf{x}^{lP}, \mathbf{x}^{\epsilon}, d_c)$, which is a function of the sensed environment $\mathbf{x}^{\epsilon}$ and threshold safety distance d_c.

The collision function δ_c is evaluated $\forall x_i \in \mathbf{x}^{lP}$ and additional points that are interpolated between the points x_i, $\mathbf{x}^{\text{itp}}$, which forms the set $\mathbf{x}^{\text{eval}} \coloneqq \mathbf{x}^{lP} \cup \mathbf{x}^{\text{itp}}$. If the path at the end of an iteration is not deemed safe ($\delta_c = \text{true}$), then the set of points $\mathbf{x}_{\text{add}} \coloneqq \{x_i^e | \forall x_i^e \in \mathbf{x}^{\text{eval}}, x_j^{\epsilon} \in \mathbf{x}^{\epsilon}, ||x_i^e, x_j^{\epsilon}|| > d_c\}$ is inserted into the path for the next iteration of the simulation, $\mathbf{x}^{lP}_{\text{iter}=n+1} \coloneqq \mathbf{x}^{lP}_{\text{iter}=n} \cup \mathbf{x}_{\text{add}}$, which is used as the initial path of the next iteration.

Figure 5 shows this iterative simulation in action, with the path eventually reaching a stable safe state, passing all safety requirements after several iterations. The evaluations points are displayed and colored by their measured safety based on a set d_c value.

IV. Results

A. Planner Validation

In order to demonstrate the capability of RVP, it is validated in simulation on randomly generated sensed environments and global paths. A large dataset of these scenarios was generated, covering a wide range of possible path-obstacle configurations with varying difficulty.

1) Dataset: Generating a random scenario begins with a random sensed environment. The sensed environment starts as an empty occupancy grid, and a random number of objects are successively placed into the map. These objects have a range of shapes and sizes and are representative of commonly occurring obstacles such as bushes and trees in an Unstructured Environment. Once the sensed environment is created, a random global path is generated. The distribution of random paths covers a wide range of scenarios, including straight paths between the start and end points as well as extremely curved paths that may intersect objects. The diversity of sensed environments and paths in the dataset allows for rigorous testing of the planner. One example scenario is shown in Figure 6 with a large number of objects and a random desired Global path.

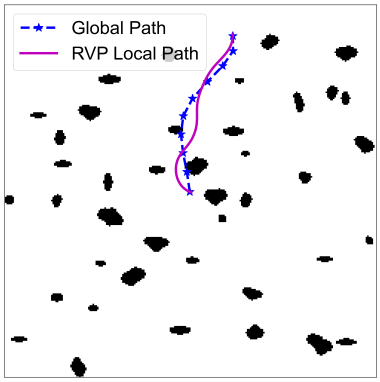
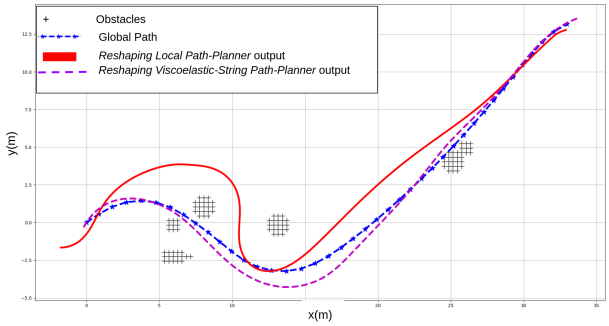


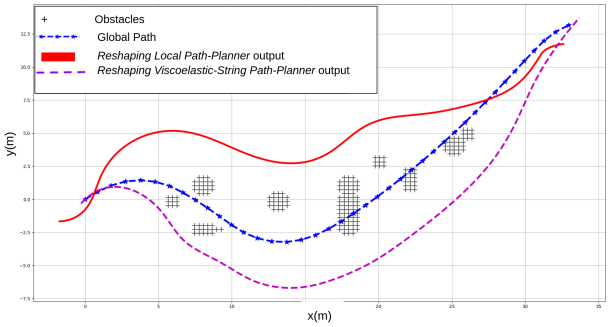
Fig. 6. *An example randomly generated sensed environment and desired Global path with its corresponding RVP reshaped Local path.*

2) Performance: Evaluating Reshaping Viscoelastic-String Path-Planner on a dataset of $10,000$ scenarios, we find that it has a success rate of $\mathbf{94.3}\%$. Here, success rate is defined as the local path $\mathbf{x}^{lP}$ having no collisions along the entire path given the sensed environment $\mathbf{x}^{\epsilon}$ i.e $!\delta_c(\mathbf{x}^{lP}, \mathbf{x}^{\epsilon})$.

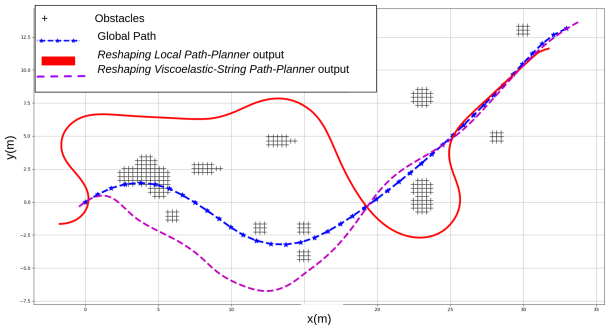
B. Comparison Scenarios between Reshaping Viscoelastic-String Path-Planner (RVP) and Reshaping Local Path Planner (RLP)



(a)



(b)



(c)

Fig. 7. *Comparison between Reshaping Local Path Planner and Reshaping Viscoelastic-String Path-Planner on 3 unstructured environments over a non-straight desired global path.*

We demonstrate the working of the RVP with a similar existing Local Planner [21] i.e *Reshaping Local Path Planner (RLP)* in Figure 7. [21] also reshapes a given Global Plan in presence of Obstacles using a *Path Agnostic Turning Moment*. In the case of the RLP, it can be seen that the RV starts at a point away from the starting point of the Global Path. Both

RLP and RVP have a look-ahead that captures the entire length of the desired Global Path.

In Figures 7(a) and 7(c), RVP performs clearly better than RLP in maintaining a small path deviation $\Delta P(\mathbf{x}^{dGP}, \mathbf{x}^{lP})$ from the desired global path. In Figure 7(b), the two planners have fairly comparable performance due to the specific path-obstacle configuration. More generally, RVP performs at least as well as RLP due to the entire path being reshaped simultaneously and the stability built into the system.

V. Conclusions and Future Work

This paper introduces a novel reshaping path planner that reshapes a Desired Global Path based on two key concepts: (i) Virtual obstacle forces that act on the path point masses and (ii) Spring-Mass-Damper system and Anchor Points, which maintain a smooth curvature and steadiness in the resultant local path.

The results of RVP are also validated on a large randomly generated dataset as well as qualitatively against a similar reshaping planner.

The key drawback of the Reshaping Viscoelastic-String Path-Planner is the computational expensiveness of approximating the ODE in equation (2). To remedy this issue, we propose training a Neural Network on a dataset of scenarios solved by RVP in order to have real-time performance. Some initial progress has been made in using a Soft Actor Critic model [26] and a Proximal Policy Optimization model [27] with a specialized reward function to learn the ODE.

References

[1] R. R. Murphy, *Disaster robotics*. MIT press, 2014.

[2] X. Xiao, B. Liu, G. Warnell, and P. Stone, "Motion control for mobile robot navigation using machine learning: a survey," 2020.

[3] L. E. Kavraki, P. Svestka, J. . Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[4] N. M. Amato and Y. Wu, "A randomized roadmap method for path and manipulation planning," in *Proceedings of IEEE international conference on robotics and automation*, vol. 1. IEEE, 1996, pp. 113–120.

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

[6] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," 2011.

[7] E. W. Dijkstra *et al.*, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[8] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[9] S. Koenig and M. Likhachev, "Dˆ* lite," *Aaai/iaai*, vol. 15, 2002.

[10] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *The International Journal of Robotics Research*, vol. 5, no. 1, pp. 90–98, 1986. [Online]. Available: https://doi.org/10.1177/027836498600500106

[11] L. Shangguan, J. A. Thomasson, and S. Gopalswamy, "Motion planning for autonomous grain carts," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 3, pp. 2112–2123, 2021.

[12] T. Weerakoon, K. Ishii, and A. A. F. Nassiraei, "An artificial potential field based mobile robot navigation method to prevent from deadlock," *Journal of Artificial Intelligence and Soft Computing Research*, vol. 5, no. 3, pp. 189–203, 2015.

[13] J. Lee, Y. Nam, S. Hong, and W. Cho, "New potential functions with random force algorithms using potential field method," *Journal of Intelligent & Robotic Systems*, vol. 66, no. 3, pp. 303–319, 2012.

[14] F. Bounini, D. Gingras, H. Pollart, and D. Gruyer, "Modified artificial potential field method for online path planning applications," in *2017 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2017, pp. 180–185.

[15] Q. Zhu, Y. Yan, and Z. Xing, "Robot path planning based on artificial potential field approach with simulated annealing," in *Sixth International Conference on Intelligent Systems Design and Applications*, vol. 2, 2006, pp. 622–627.

[16] N. Hirose, F. Xia, R. Martin-Martin, A. Sadeghian, and S. Savarese, "Deep visual mpc-policy learning for navigation," *IEEE Robotics and Automation Letters*, vol. 4, pp. 3184–3191, 10 2019.

[17] Y. Pan, C.-A. Cheng, K. Saigol, K. Lee, X. Yan, E. Theodorou, and B. Boots, "Agile autonomous driving using end-to-end deep imitation learning," 2019.

[18] J. Ji, A. Khajepour, W. W. Melek, and Y. Huang, "Path planning and tracking for vehicle collision avoidance based on model predictive control with multiconstraints," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 2, pp. 952–964, 2017.

[19] J. Zhang, J. T. Springenberg, J. Boedecker, and W. Burgard, "Deep reinforcement learning with successor features for navigation across similar environments," 2017.

[20] O. Zhelo, J. Zhang, L. Tai, M. Liu, and W. Burgard, "Curiosity-driven exploration for mapless navigation with deep reinforcement learning," 2018.

[21] A. Sarvesh, A. Carroll, and S. Gopalswamy, "Reshaping local path planner," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 6534–6541, 2022.

[22] S. Siva, M. Wigness, J. Rogers, and H. Zhang, "Robot adaptation to unstructured terrains by joint representation and apprenticeship learning," in *Robotics: Science and Systems (RSS)*, 2019.

[23] U. Muller, J. Ben, E. Cosatto, B. Flepp, and Y. L. Cun, "Off-road obstacle avoidance through end-to-end learning," in *Advances in neural information processing systems*. Citeseer, 2006, pp. 739–746.

[24] M. Wigness, J. G. Rogers, and L. E. Navarro-Serment, "Robot navigation from human demonstration: Learning control behaviors," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 1150–1157.

[25] S. Quinlan and O. Khatib, "Elastic bands: Connecting path planning and control," in *[1993] Proceedings IEEE International Conference on Robotics and Automation*. IEEE, 1993, pp. 802–807.

[26] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International conference on machine learning*. PMLR, 2018, pp. 1861–1870.

[27] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.