Gaussian Random Paths for Real-Time Motion Planning

Sungjoon Choi, Kyungjae Lee, and Songhwai Oh

Abstract— **In this paper, we propose Gaussian random paths by defining a probability distribution over continuous paths interpolating a finite set of anchoring points using Gaussian process regression. By utilizing the generative property of Gaussian random paths, a Gaussian random path planner is developed to safely steer a robot to a goal position. The Gaussian random path planner can be used in a number of applications, including local path planning for a mobile robot and trajectory optimization for a whole body motion planning. We have conducted an extensive set of simulations and experiments, showing that the proposed planner outperforms look-ahead planners which use a pre-defined subset of egocentric trajectories in terms of collision rates and trajectory lengths. Furthermore, we apply the proposed method to existing trajectory optimization methods as an initialization step and demonstrate that it can help produce more cost-efficient trajectories.**

I. Introduction

The notion of a *trajectory* has been extensively used in the field of robotics. In the context of autonomous navigation, a trajectory of an autonomous vehicle is often generated while preventing collision with obstacles. On the other hand, in the context of a manipulator or humanoid robot, a trajectory in the configuration space is often optimized to make the end-effector move to the desired location while guaranteeing the feasibility and safety.

In this regard, sampling-based techniques, such as rapidly exploring random tree (RRT) [1] or probabilistic roadmap (PRM) [2], have been proposed. While these randomized search based methods can guarantee *probabilistic completeness*, the notion of *cost* or *optimality* cannot be directly incorporated in this manner. To handle this issue, an additional optimization step is added after finding a feasible trajectory. Recently proposed algorithms, such as RRT* [3], can guarantee both *completeness* as well as *asymptotic optimality*.

However, randomized search based methods often require a large amount of time to find the optimal or even a feasible path once the dimension of the trajectory increases. RRT*, for example, has to expand nodes to cover the entire space to find the optimal trajectory. Due to this limitation, trajectory optimization methods have been widely used for complex motion planning problems. Sampling or gradient based optimization techniques, e.g., CHOMP [4], STOMP [5], and TrajOpt [6], are utilized to find a locally optimal path given an initial trajectory. One drawback of trajectory optimization methods is that their performance is sensitive to the given initial trajectory.

In this paper, we focus on the problem of sampling diverse trajectories that passes through given anchoring points (waypoints) and propose Gaussian random paths (GRPs) by defining a probability distribution over continuous paths using a Gaussian process. Using the fact that sampled GRPs connect certain waypoints, e.g., the current position of a robot and its goal position, a Gaussian random path planner is proposed and applied to local path planning.

Furthermore, the proposed path set generation scheme is also applied to a high-dimensional trajectory optimization problem such as a whole body motion planning of a humanoid robot. From a comprehensive set of experiments, we show that the trajectory optimization initialized with GRPs can find more cost-efficient trajectories, compared to the linearly interpolated initialization method, which is often used in practice. In addition, the GRP initialization can significantly reduce the time required by the trajectory optimization routine.

The remainder of this paper is organized as follows. The proposed Gaussian random path is defined in Section II along with the learning algorithm for GRP given a specific dynamic model and the ϵ run-up method. In Section III, a local path planner using GRPs is described. We also demonstrate how GRPs can be used to improve the quality and speed of existing trajectory optimization methods in Section IV.

II. Gaussian Random Paths

In order to describe Gaussian random paths, we first define a path and anchoring points.

Definition 1: A path $\mathbf{p}$ is a vector-valued function where the input space is a time interval $\mathcal{I} \subseteq \mathbb{R}$ and the output space is a set of locations:

$$\mathbf{p} : \mathcal{I} \rightarrow \mathbb{R}^d,$$

where d is the dimension of the space.

Definition 2: Anchoring points $D_a = (\mathbf{s}_a, \mathbf{x}_a)$ are a set, which consists of M time indices $\mathbf{s}_a = \{s_i | i = 1, 2, ..., M\}$ and locations $\mathbf{x}_a = \{\mathbf{x}_i | i = 1, 2, ..., M\}$, such that a path must pass through.

In order to sample a path, we treat a path defined in Definition 1 as a random process, namely a Gaussian process, and define a probability distribution over continuous paths. A Gaussian process is a generalization of the Gaussian probability distribution. Whereas a probability distribution describes random variables which are scalar or vectors, a stochastic process governs the properties of functions. While

This work was supported by the Institute for Information & Communications Technology Promotion (IITP) grant funded by the Korea government (MSIP) (B0101-16-0307, Basic Software Research in Human-Level Lifelong Machine Learning).

S. Choi, K. Lee, and S. Oh are with the Department of Electrical and Computer Engineering and ASRI, Seoul National University, Seoul 151-744, Korea (e-mail: {sungjoon.choi, kyungjae.lee, songhwai.oh}@cpslab.snu.ac.kr).

dealing with infinite dimensional objects is computationally infeasible in most cases, the marginalization property of the Gaussian distribution makes it tractable [7]. This property is particularly important in modeling paths since a path modeled by a Gaussian process is free from quantization issues.

Unfortunately, a Gaussian process itself is not suitable for directly modeling a path which passes through certain anchoring points and, to achieve this, we revisit Gaussian process regression (GPR). GPR computes a Gaussian posterior distribution and we can use this GPR framework to model a distribution of paths with anchoring points, where the training data corresponds to the anchoring points. Now, we are ready to define Gaussian random paths.

Definition 3: Given a kernel function $k(t, t')$ and a set of M anchoring points $D_a = (\mathbf{s}_a, \mathbf{x}_a) = \{(s_i, x_i) \,|\, i = 1, 2, \dots, M\}$, a Gaussian random path $\mathcal{P}$ with a sequence of T test time indices $\mathbf{t}_{test} = \{t_i \,|\, i = 1, 2, \dots, T\}$ is specified with a mean path $\mu_{\mathcal{P}}$ and a covariance matrix $K_{\mathcal{P}}$.

$$\mathcal{P} \sim \mathcal{N}(\mu_{\mathcal{P}}, K_{\mathcal{P}}), \tag{1}$$

where

$$\mu_{\mathcal{P}} = \mathbf{k}(\mathbf{t}_{test}, \mathbf{s}_a)^T (\mathbf{K}_a + \sigma_w^2 I)^{-1} \mathbf{x}_a, \tag{2}$$

$$\mathcal{K}_{\mathcal{P}} = \mathbf{K}_{test} - \mathbf{k}(\mathbf{t}_{test}, \mathbf{s}_a)^T (\mathbf{K}_a + \sigma_w^2 I)^{-1} \mathbf{k}(\mathbf{t}_{test}, \mathbf{s}_a), \tag{3}$$

$\mathbf{k}(\mathbf{t}_{test}, \mathbf{s}_a) \in \mathbb{R}^{T \times M}$ is a kernel matrix of test time indices and anchoring time indices, $\mathbf{K}_a = \mathbf{K}(\mathbf{s}_a, \mathbf{s}_a) \in \mathbb{R}^{M \times M}$ is a kernel matrix of anchoring time indices, and $\mathbf{K}_{test} = \mathbf{K}(\mathbf{t}_{test}, \mathbf{t}_{test}) \in \mathbb{R}^{T \times T}$ is a kernel matrix of test time indices. Note that this is an one-dimensional path case and the model can easily be extended to a d-dimensional path by assuming independence between each dimension.

The proposed Gaussian random path (GRP) differs from usual Gaussian process regression in that, whereas the main purpose of regression is to predict an output of an unseen input, the GRP focuses on defining a distribution over a sequence of time indices in the interval $\mathcal{I}$ and sampling paths interpolating a set of anchoring points from the distribution.

Another useful property of a Gaussian process is that its realization is C^n continuous, if $2n$ moments of the scale parameter exist [8]. While computing the n-th moment of the scale parameter can be demanding, it is known that a realization of a Gaussian process with a squared exponential and rational quadratic kernel function are infinitely differentiable. This property is particularly important in path planning since we often require smooth paths satisfying certain constraints.

Proposition 1: Paths sampled from the GRP distribution with a squared exponential kernel function are C^∞ continuous.

Proof: The proof comes directly from the definition of a path in Definition 1. A path is defined as a vector-valued function of time, and thus, a realization of a Gaussian process is equivalent to sampling a path from the Gaussian random path distribution. ■

The proposed Gaussian random paths can be effectively used in sampling-based trajectory optimization algorithms, e.g., STOMP [5]. While STOMP requires an additional smoothing step as it utilizes noisy sampled trajectories, paths sampled from the Gaussian random path planner in Section III are already smooth and thus an additional step is not required. Moreover, since the GRP distribution can incorporate any arbitrary anchoring points into account, it can be applied to the problem of optimizing a trajectory over waypoints.

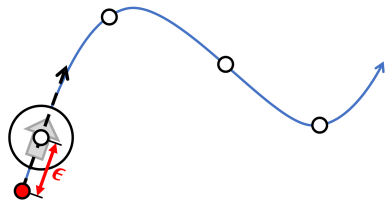


Fig. 1: An ϵ run-up method applied to GRPs with three anchoring points. An added extra anchoring point is shown in a red disk.

A. ϵ Run-Up Method

When it comes to applying GRPs to a unicycle model, sampled paths may not directly be used by a low level controller due to its nonholonomic constraints. In other words, the resulting path should be *trackable* with respect to the specific dynamic model.

Definition 4: A path $\mathbf{p}(t) = [\mathbf{p}_x(t) \;\; \mathbf{p}_y(t)]^T$ is *trackable* if the gradient of a path $[\frac{d\mathbf{p}_x(t)}{dt} \;\; \frac{d\mathbf{p}_y(t)}{dt}]^T$ at t_0 equals to the heading vector $[v \cdot \cos(\theta_0) \;\; v \cdot \sin(\theta_0)]^T$ of a robot, where t_0 is the time index of a robot at its origin, v is the directional velocity, and $\theta_0 \in [0, 2\pi)$ is the heading of a robot.

This problem can be handled by the ϵ run-up method proposed below. Suppose that the position, heading, and directional velocity of a robot are $\mathbf{x}_0 = [x_0 \;\; y_0]^T$, $\theta_0 \in [0, \, 2\pi)$, and v, respectively. Then the ϵ run-up method adds an additional anchoring point $[t_{ru} \;\; \mathbf{x}_{ru}]^T$ to the original set of anchoring points as shown in Figure 1, where

$$t_{ru} = t_0 - \epsilon / v \tag{4}$$

and

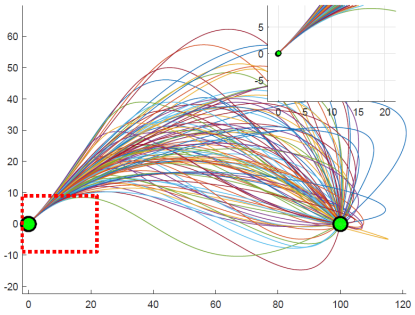
$$\mathbf{x}_{ru} = [x_0 - \epsilon \cdot \cos(\theta_0) \;\;\; y_0 - \epsilon \cdot \sin(\theta_0)]^T. \tag{5}$$

If a sampled path is C^1 continuous, then the path becomes *trackable* as ϵ goes to zero.

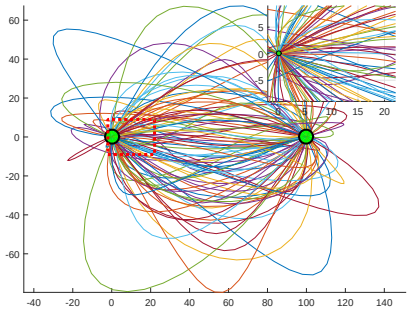
Proposition 2: Paths sampled from the GRP distribution with a squared exponential kernel function with the ϵ run-up method are *trackable* as ϵ goes to zero.

Proof: Let $\mathbf{x}_0$ and θ_0 be the position and heading of a robot, respectively, and $\mathbf{p}$ be a sampled path from the GRP distribution with a squared exponential kernel function given an ϵ run-up anchoring point. Then $[\frac{\epsilon \cdot \cos(\theta_0)}{\epsilon / v} \;\; \frac{\epsilon \cdot \sin(\theta_0)}{\epsilon / v}]^T$ corresponds to the left-derivative of $\mathbf{p}$ at t_0. The path $\mathbf{p}$ is infinitely differentiable at $\mathbf{x}_0$ from Proposition 1 meaning that the left derivative at $\mathbf{x}_0$ equals to the right derivative, which completes the proof. ■

Figure 2 shows sampled paths with and without the ϵ run-up method where the heading at the origin is $45°$.



(a)



(b)

Fig. 2: Paths sampled from the GRP distribution (a) with ϵ run-up (b) without ϵ, where the heading of a robot is $45°$.

III. Local Path Planner

In many cases, a mobile robot is ordered to accomplish assigned tasks by navigating through given waypoints. For instance, in the DARPA Urban Challenge, the objective of a path planner was to find a collision-free local path connecting given waypoints using sensory measurements [10].

The proposed Gaussian random path can be effectively applied to a local planning problem. Given the current position of a robot and the goal position (waypoint), we first sample a diverse set of trackable paths from the GRP distribution with the ϵ run-up method from Section II-A, where ϵ is set to 100mm. Then an occupancy grid map constructed from the current range sensor measurements is used to examine the sampled paths and the algorithm selects the shortest collision free path with respect to the occupancy grid map. We used a pure-pursuit algorithm [11]

Furthermore, we utilized Graphics processing units (GPUs) to alleviate the computational load by assigning a GPU thread to each point in a path. In particular, we used an NVIDIA GTX870m graphics card with 1344 CUDA cores and 6GB of memory and achieved a considerable speedup, which requires less than 10ms to examine 500 paths. An overall path planning algorithm is summarized in Algorithm 1 to control the robot to follow the shortest collision-free path. The whole control process including the generation of 500 paths and examination took about 10ms in MATLAB. We would like to note that this speedup using GPU is only related to examining the paths with respect to current occupancy configurations which is used in both the path planning using the Gaussian random path and look-ahead planner.

Algorithm 1 Gaussian Random Path Planner (GRPP)

1: **Input** An occupancy grid map M, a robot position $\mathbf{x}_0$, a directional velocity v, and a goal position $\mathbf{x}_g$
2: **Output** A Gaussian random path $\mathbf{p}_{opt}$
3: Sample a set $\mathcal{P}$ of paths from the GRP distribution with anchoring points $D_a = (\mathbf{s}_a, \mathbf{x}_a)$ where $\mathbf{s}_a = \{t_{ru}, t_0, t_g\}$, $\mathbf{x}_a = \{\mathbf{x}_{ru}, \mathbf{x}_0, \mathbf{x}_g\}$, $t_0 = 0$, and $t_g = ||\mathbf{x}_g - \mathbf{x}_0||/v$.
4: Find the shortest collision-free path $\mathbf{p}_{opt}$ among $\mathcal{P}$ with respect to M.
5: Control the robot to follow $\mathbf{p}_{opt}$ using a pure-pursuit algorithm.

A. Simulation Study

To validate the performance of the proposed method, a Gaussian random path planner (GRPP), we compared our method with a look-ahead planner (LAP) [10], [12]–[14], which utilizes a path set method where the look-ahead time of a path set varies from 2 to 10 seconds. For a fair comparison, $1,000$ paths are pre-computed offline in each path set as this setting makes the computing time of each path planner similar as shown in Table I.

We assume that a robot can only obtain information from its egocentric view with a $120°$ field of view to make a realistic setup in that a decision is made solely based on the occupancy grid map generated from a partially observable view. For each run, the goal is to reach the target position located at $(100, 0)$ and the number of obstacles varies from 2 to 12 and they are randomly placed. For each setting, 50 independent runs are performed and collision rates as well as the total moved distance is computed. Excluding the path generation step, the LAP works identical to the GRPP.

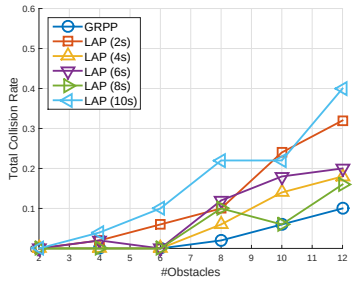
The quantitative simulation results are shown in Figure 3. In Figure 3(a), the total collision rate is presented. In most cases, the GRPP outperforms the LAPs and this is mainly due to the fact that whereas the GRPP considers a full path connecting the current robot position to the target position, the LAP only considers its look-ahead period.

The total moved distance is shown in Figure 3(b) and we can see that the GRPP achieves shorter moving distances. The inherent drawback of the LAP in that it is sensitive to the look-ahead period. The LAP with a shorter look-ahead period makes a myopic control leading to a higher direct collision rate. In contrast, as the look-ahead period gets longer, the probability of entering an inevitable collision state [15], where all paths are obstructed by obstacles, increases. This phenomenon is shown in Figure 3(c).

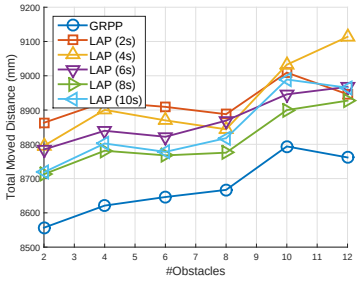
[1]An inevitable collision state is the condition when all paths are blocked by obstacles [15] and a direct collision indicates the situation when a robot collides with an obstacle while executing the current control.

	GRPP	LAP (2s)	LAP (4s)	LAP (6s)	LAP (8s)	LAP (10s)
Sampling Time (ms)	0.82 (0.037)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)
Examine Time (ms)	9.84 (0.13)	10.08 (0.11)	10.06 (0.12)	10.04 (0.12)	10.01 (0.09)	10.01 (0.13)
Total Time (ms)	10.66 (0.14)	10.08 (0.11)	10.06 (0.12)	10.04 (0.12)	10.01 (0.09)	10.01 (0.13)

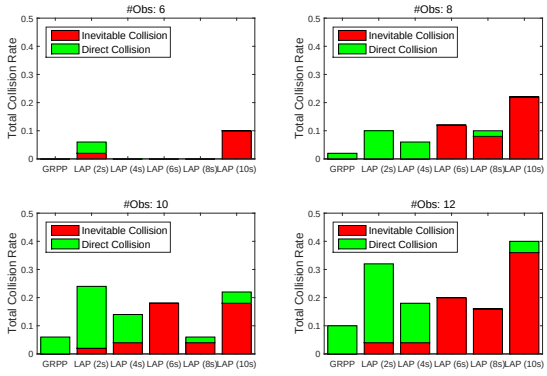
TABLE I: Computation times of tested path planners: GRPP with 500 paths and LAPs with $1,000$ paths. The numbers in parentheses refer to the standard deviation.



(a)



(b)



(c)

Fig. 3: Local path planning simulation results of the proposed GRPP and look-ahead planners (LAPs) with different prediction periods (2, 4, 6, 8, and 10 seconds): (a) total collision rates, (b) total moved distances, and (c) inevitable collision and direct collision rates[1].

B. Experiment

We conducted local path planning experiments using a Pioneer 3DX mobile robot with two mounted Microsoft Kinect cameras. The objective is to navigate safely to the goal position 4m away in two different wall configurations, a convex wall and a concave wall. While path planning in a convex wall is relatively easy, the horseshoe-shaped wall configuration in a concave wall makes navigation hard as a robot is more likely to enter an inevitable collision state.

		Convex wall	Concave wall
GRP	collision rate	0%	0%
	moving distance	4477.8 (182.0)	4537.4 (233.6)
LAP (8s)	collision rate	0%	21.4%
	moving distance	4583.6 (440.7)	4317.9 (170.0)

TABLE II: Average collision rates and moving distances (mm) of GRPP and LAP in two wall configurations. The numbers in parentheses refer to the standard deviation.

We compared the proposed Gaussian random path planner (GRPP) with the look-ahead planner (LAP) with 8 seconds look-ahead time as it showed the best performance among other LAPs. For each scenario, 10 independent experiments are performed and the quantitative results are shown in Table II.

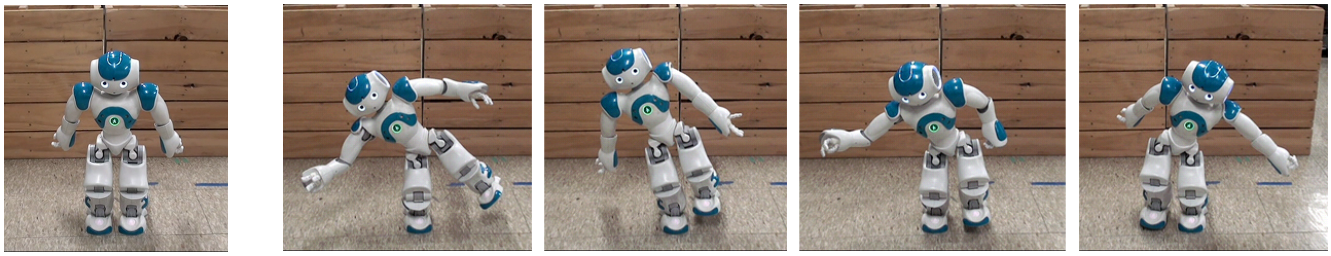
In the easier case (convex wall), both the GRPP and the LAP successfully control the robot to the goal position and the average moved distance of the GRPP is slightly shorter than that of the LAP. On the other hand, for the concave wall case, the LAP shows a collision rate of 21%, whereas the GRPP achieves collision-free navigation in all cases.

IV. Trajectory Optimization Initialization

It is well known that for robotic motion planning with many degree of freedom (DOF), trajectory optimization techniques such as CHOMP [4], STOMP [5], cross-entropy (CE) motion planning [16] or TrajOpt [6], are more appropriate compared to randomized sampling based algorithms, such as RRT or RRT* [3], in terms of time and memory complexities [16]. Given an initial trajectory, a trajectory optimization algorithm gradually updates the trajectory such that the predefined cost function decreases. Despite the successes in a number of applications, a trajectory optimization algorithm suffers from its major drawback in that it heavily depends upon the initial trajectory [17]. In [17], Pan et. al. proposed a learning-based approach to predict the quality of an initial trajectory.

We apply the Gaussian random path planning algorithm to find an appropriate initial trajectory for a trajectory optimization problem and compare with the initialization using linear interpolation which is often applied in practice [5], [6]. In particular, we applied GRP to three widely used trajectory optimization methods: CHOMP [4], STOMP [5], and cross-entropy (CE) motion planning [16].

The objective of the experiment considered in this paper is to find a whole body trajectory of a humanoid robot



(a) (b)

Fig. 4: Trajectory optimization for a humanoid robot, Nao. (a) An example of an initial pose. (b) Examples of extreme final poses.

	Example 1	Example 2	Example 3	Example 4	Example 5	Example 6	Example 7	Example 8
GRP (10)	214.10 (4.28)	227.23 (4.34)	148.95 (3.48)	227.50 (4.35)	125.42 (3.53)	227.50 (4.35)	161.52 (4.03)	161.52 (4.03)
GRP (200)	208.00 (1.95)	221.04 (1.95)	141.31 (2.50)	221.27 (1.96)	117.93 (2.35)	221.27 (1.96)	155.38 (2.28)	155.38 (2.28)
GRP (500)	204.91 (2.20)	218.02 (2.19)	138.70 (2.42)	218.23 (2.20)	115.66 (2.30)	218.23 (2.20)	151.96 (2.53)	151.96 (2.53)
CHOMP (LI)	0.24 (0.00)	0.88 (0.00)	37.78 (0.00)	**13.09 (0.00)**	22.96 (0.00)	**13.09 (0.00)**	**3.00 (0.00)**	**3.00 (0.00)**
STOMP (LI)	260.23 (27.43)	256.68 (27.46)	54.18 (24.58)	227.39 (26.59)	51.43 (23.63)	227.39 (26.59)	244.60 (26.04)	244.60 (26.04)
CE (LI)	202.31 (9.62)	199.00 (10.04)	34.21 (15.04)	174.85 (9.77)	33.48 (16.15)	174.85 (9.77)	189.98 (10.17)	189.98 (10.17)
CHOMP (GRP)	**0.12 (0.00)**	**0.77 (0.03)**	37.80 (2.44)	13.96 (1.83)	**21.41 (1.67)**	13.96 (1.83)	3.14 (0.52)	3.14 (0.52)
STOMP (GRP)	67.49 (22.41)	67.66 (22.09)	58.25 (24.06)	67.49 (21.99)	56.19 (23.99)	67.49 (21.99)	64.25 (21.93)	64.25 (21.93)
CE (GRP)	42.43 (8.61)	42.41 (8.30)	**33.25 (12.04)**	42.05 (8.32)	32.94 (11.88)	42.05 (8.32)	40.12 (9.58)	40.12 (9.58)
CHOMP Impv.	49.22 %	13.14 %	-0.05 %	-6.21 %	6.73 %	-6.21 %	-4.27 %	-4.27 %
STOMP Impv.	74.07 %	73.64 %	-6.99 %	70.32 %	-8.46 %	70.32 %	73.73 %	73.73 %
CE Impv.	79.03 %	78.69 %	2.80 %	75.95 %	1.61 %	75.95 %	78.88 %	78.88 %

TABLE III: Simple trajectory optimization results. The numbers in parentheses refer to the standard deviation.

	Example 1	Example 2	Example 3	Example 4	Example 5	Example 6	Example 7	Example 8
GRP (10)	78.83 (26.96)	120.52 (27.54)	112.45 (47.30)	50.14 (15.67)	77.13 (24.12)	89.46 (28.04)	126.53 (54.60)	125.92 (39.11)
GRP (200)	26.99 (4.87)	45.19 (15.79)	52.20 (12.57)	18.14 (7.37)	32.81 (5.49)	36.92 (7.99)	52.93 (12.19)	51.10 (13.16)
GRP (500)	23.66 (2.61)	32.61 (5.07)	44.07 (12.25)	11.86 (2.93)	26.93 (3.44)	28.55 (3.46)	46.05 (10.82)	39.55 (10.00)
CHOMP (LI)	446.62 (0.00)	105.20 (0.00)	93.25 (0.00)	32.61 (0.00)	239.88 (0.00)	274.99 (0.00)	190.27 (0.00)	105.19 (0.00)
STOMP (LI)	266.99 (10.55)	197.07 (18.48)	169.12 (15.28)	335.40 (38.03)	246.03 (13.93)	276.77 (21.26)	175.60 (16.03)	195.42 (18.29)
CE (LI)	232.11 (6.23)	112.92 (6.81)	97.13 (7.50)	207.46 (22.59)	209.62 (7.09)	227.12 (9.80)	102.90 (8.03)	111.52 (8.50)
CHOMP (GRP)	23.66 (2.61)	**1.99 (0.01)**	**7.46 (2.58)**	11.54 (2.29)	26.93 (3.44)	28.55 (3.46)	45.17 (10.71)	**1.99 (0.00)**
STOMP (GRP)	21.65 (3.24)	22.65 (5.52)	23.79 (6.31)	**11.46 (3.16)**	**22.11 (3.58)**	**22.60 (4.08)**	**23.90 (3.50)**	25.24 (3.17)
CE (GRP)	**21.50 (2.23)**	24.25 (6.30)	24.79 (8.14)	11.86 (2.93)	24.11 (3.02)	23.27 (3.11)	28.04 (4.77)	27.70 (6.79)
CHOMP Impv.	94.70 %	98.11 %	92.00 %	64.61 %	88.77 %	89.62 %	76.26 %	98.11 %
STOMP Impv.	91.89 %	88.51 %	85.93 %	96.58 %	91.01 %	91.84 %	86.39 %	87.09 %
CE Impv.	90.74 %	78.53 %	74.48 %	94.29 %	88.50 %	89.76 %	72.75 %	75.16 %

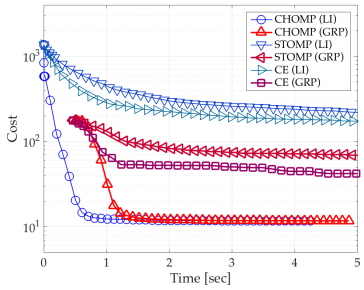
TABLE IV: Extreme trajectory optimization results. The numbers in parentheses refer to the standard deviation.

connecting two configurations while satisfying the stability constraints, which can be expressed as

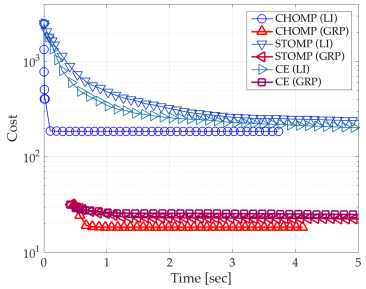
$$\begin{aligned} \underset{\bar{\theta}}{\text{minimize}} \quad & \textstyle\sum_{r=1}^{n-1} \|\theta_{r+1} - \theta_r\|_W^2 \\ \text{subject to} \quad & \theta_{min} \leq \theta_r \leq \theta_{max}, r = 1, \ldots, n \\ & X_r = X_{CoM}(\theta_r), r = 1, \ldots, n, \end{aligned}$$

where $\theta_r \in \mathbb{R}^{23}$ is the r-th pose in the trajectory $\bar{\theta}$, W is a diagonal matrix indicating weights, $\|\theta_i - \theta_j\|_W^2$ is a weighted distance defined as $(\theta_i - \theta_j)^T W(\theta_i - \theta_j)$ and the weight matrix W is proportional to the mass of the joint, X_r is the desired center of mass (CoM), θ_{min} and θ_{max} are the range of joint angles, and $X_{CoM}(\theta)$ is a CoM of θ. The augmented Lagrangian multiplier method is used to find the solution $\bar{\theta}^*$, which is similar to the formulation in [4]. Each pose consists of 23 joints excluding both wrist joints.

We conducted two sets of experiments using a humanoid robot, Nao. In the first set, a robot changes its configuration from its initial pose to eight different simple poses. In the second set, a robot changes its configuration to stand with one foot. Again, eight different final poses are tested and we call them extreme final poses. Figure 4 shows an initial and four extreme final poses considered in experiments. Standing with only one foot reduces the stable region of the CoM of a humanoid robot, i.e., the supporting plane of one foot, making the trajectory optimization more challenging. Table III and IV summarize the results, including the initial costs of trajectories generated by GRP with 10, 200, and 500 sampled trajectories. The tables show final costs of trajectories computed by three trajectory optimization methods, CHOMP, STOMP, and CE, where each algorithm is initialized with linear interpolation (LI) or GRP. We can clearly see the benefit of using GRP as initialization compared to linear interpolation. The cost improvements in percentage are also shown for all methods. For each case, we have conducted 10 independent runs and the average cost values are shown in the tables.



(a)



(b)

Fig. 5: Average costs as a function of algorithm's execution time. (a) Simple trajectory optimization. (b) Extreme trajectory optimization. Note the log scale for cost (y-axis).

For simple trajectory optimization cases, the GRP initialization results in performance improvement for CE and STOMP while the GRP initalization has not shown an improvement for CHOMP. For extreme cases, however, the GRP initialization improves the performance of all trajectory optimization methods. In fact, in all extreme cases, the initial cost of the 500 sampled trajectories with the GRP outperforms the final costs of CHOMP, STOMP, and CE with linear interpolation. It shows that GRP is quite effective in finding a good initial trajectory for complex problems compared to linear interpolation.

The average costs for the tested algorithms as a function of time (sec) are shown in Figure 5. [2] Since GRP takes 0.46 seconds to sample and evaluate 500 trajectories, the methods with the GRP initialization starts at time 0.46 seconds. As illustrated in Figure 5(a), excluding CHOMP, GRP effectively improves the performance of CE and STOMP. Note that the performance gap between CHOMP with linear interpolation and CHOMP with GRP is neglectable. However, for the extreme cases, GRP significantly improves performance of all trajectory optimization methods.

V. Conclusion

In this paper, we have proposed a rapid path sampling method, called Gaussian random paths, to generate a diverse set of paths passing through anchoring points using Gaussian process regression. The Gaussian random path planner is proposed by exploiting the efficiency of Gaussian random paths. By fully utilizing GPUs, the overall process of generating 500 random paths and finding the optimal feasible path takes about 10ms in MATLAB. In both simulations and experiments, the proposed GPR planner outperforms a look-ahead planner in all cases in terms of collision rates and trajectory lengths. We have also shown that the proposed Gaussian random path can be effectively used as an initialization step for widely used trajectory optimization methods such as CHOMP, STOMP, and CE.

[2] All programs are written in mex-compiled MATLAB and a 3.2GHz quad-core processor is used.

References

[1] J. J. Kuffner and S. M. LaValle, "RRT-connect: An efficient approach to single-query path planning," in *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, April 2000.

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

[3] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.

[4] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa, "CHOMP: Gradient optimization techniques for efficient motion planning," in *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, May 2009.

[5] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, "STOMP: Stochastic trajectory optimization for motion planning," in *Proc. of the IEEE International Conference of Robotics and Automation (ICRA)*, May 2011.

[6] J. Schulman, J. Ho, A. Lee, I. Awwal, H. Bradlow, and P. Abbeel, "Finding locally optimal, collision-free trajectories with sequential convex optimization." in *Proc. of Robotics: Science and Systems (RSS)*, June 2013.

[7] C. Rasmussen and C. Williams, *Gaussian processes for machine learning*. MIT press Cambridge, MA, 2006, vol. 1.

[8] C. J. Paciorek, "Nonstationary Gaussian processes for regression and spatial modelling," Ph.D. dissertation, Carnegie Mellon University, 2003.

[9] M. K. C. Tay and C. Laugier, "Modelling smooth paths using Gaussian processes," in *Field and Service Robotics (Springer Tracts in Advanced Robotics, vol. 42)*. New York, NY, USA: Springer-Verlag, 2008, pp. 381–390.

[10] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann *et al.*, "Stanley: The robot that won the DARPA Grand Challenge," *Journal of field Robotics*, vol. 23, no. 9, pp. 661–692, 2006.

[11] O. Amidi and C. E. Thorpe, "Integrated mobile robot control," Robotics Institute, Carnegie Mellon University, Tech. Rep. Robotics Institute, Carnegie Mellon University, 1990.

[12] J. Bohren, T. Foote, J. Keller, A. Kushleyev, D. Lee, A. Stewart, P. Vernaza, J. Derenick, J. Spletzer, and B. Satterfield, "Little Ben: the Ben Franklin racing team's entry in the 2007 DARPA Urban Challenge," *Journal of Field Robotics*, vol. 25, no. 9, pp. 598–614, 2008.

[13] D. Ferguson, T. M. Howard, and M. Likhachev, "Motion planning in urban environments," *Journal of Field Robotics*, vol. 25, no. 11-12, pp. 939–960, 2008.

[14] A. Bacha, C. Bauman, R. Faruque, M. Fleming, C. Terwelp, C. Reinholtz, D. Hong, A. Wicks, T. Alberi, D. Anderson *et al.*, "Odin: Team VictorTango's entry in the DARPA Urban Challenge," *Journal of Field Robotics*, vol. 25, no. 8, pp. 467–492, 2008.

[15] T. Fraichard and H. Asama, "Inevitable collision states – a step towards safer robots?" *Advanced Robotics*, vol. 18, no. 10, pp. 1001–1024, 2004.

[16] M. Kobilarov, "Cross-entropy motion planning," *International Journal of Robotics Research*, vol. 31, no. 7, pp. 855–871, 2012.

[17] J. Pan, Z. Chen, and P. Abbeel, "Predicting initialization effectiveness for trajectory optimization," in *Proc. of the IEEE International Conference of Robotics and Automation (ICRA)*, June 2014.