Proceedings of 2019 IEEE
International Conference on Mechatronics and Automation
August 4 - 7, Tianjin, China

A Motion Planning Algorithm Based on Trajectory Optimization with Workspace Goal Region

Kai Mi[1,2] and Peng Hao[1,2]

1. University of Chinese Academy of Sciences

University of Chinese Academy of Sciences
Yuquan Road 19, Beijing, China
{mikai2015}@ia.ac.cn

Jun Zheng[2], Yunkuan Wang[2] and Jianhua Hu[2]

2. Intelligent Manufacturing Technology and System Research Center
Institute of Automation Chinese Academy of Sciences
Zhongguancun East Road 95, Beijing, China
{jun.zheng}@ia.ac.cn

***Abstract* - We consider a motion planning problem with workspace goal region in obstacle environments and the task is to move the end-effector into a specific goal region without posture constraints. In order to improve the quality of the planning trajectory, we present a goal region constraint algorithm based on Gaussian process motion planning. We construct a distance field for the irregular goal region in advance. The closest distance and direction from any location of the workspace to the specific goal region can be easily computed. Combined with the original method, we define a goal-region-constrained likelihood which specifies the probability that the position of end-effector is within the specific goal region and move the end-effector to a better position by numerical optimization. Finally, multiple simulation experiments are carried out and the results show that the proposed algorithm can quickly plan an obstacle avoidance trajectory in joint space and the quality of the trajectory is improved effectively compared to randomly specifying a goal configuration.**

Index Terms –Workspace goal region, Distance field, Goal-region-constrained likelihood, Trajectory optimization, Obstacle avoidance.

I. INTRODUCTION

Many practical manipulation tasks afford a large amount of freedom in the choice of the goal locations. For example, when we throw an object into the recycling bin as is shown in Fig. 1. We can choose a random posture, as well as a wide range of goal locations above the recycling bin to release the object. Then the different feasible goal locations make up a workspace goal region. For an irregularly shaped box, the corresponding goal region is also irregular. The task is to move the end-effector into the goal region, meanwhile avoiding obstacles is essential.

Obstacle avoidance motion planning is one of the key technologies for robotic self-programming. The purpose is to find a smooth trajectory through the robot's configuration space and the trajectory is collision-free and obeys the robot's physical limitations. The traditional grid search methods such as the A* algorithm [1] or D* algorithm [2] are not suitable for high degree-of-freedom (DOF) manipulators, because their computational complexity will increase exponentially with the increase of the configuration space dimension. Currently, sampling-based algorithms and trajectory optimization approaches are two main types of motion planning planners for high DOF robots.

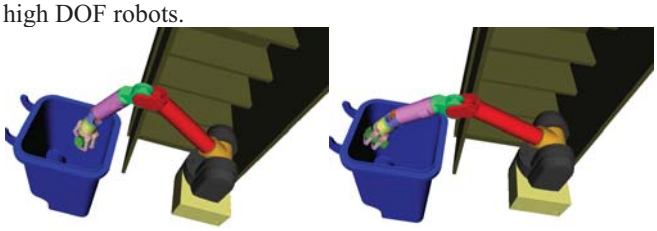

Fig. 1: Snapshots of throwing an object in different poses and positions with the WAM arm in a simulated environment.

Sampling-based approaches have become popular due to the probability completeness and they can easily plan a valid path for high degree-of-freedom (DOF) systems. The most basic sampling-based algorithms are the probabilistic roadmap (PRM) [3] and rapidly exploring random trees (RRT) [4]. Many improved algorithms [5] [6] have also been proposed. In general, the sampling-based algorithm is very efficient and probabilistically complete. However, compared to the trajectory optimization approaches, these algorithms need a longer time to generate an optimized trajectory. In order to plan a feasible and optimal trajectory, some trajectory optimization methods have been proposed. For example, the CHOMP planner [7] optimizes an initial trajectory using covariant gradient descent. For problems with non-differentiable constraints, STOMP planner [8] samples a series of noisy trajectories to explore the space around an initial trajectory. All of the above methods optimize a discrete initial trajectory and a fine discretization is needed to reason about obstacles in complex environment. Then the computational cost will increase significantly. To overcome this problem, Gaussian Process Motion Planner (GPMP) [9] samples a few states on the initial trajectory and uses Gaussian process (GP) interpolation to query the trajectory at any time of interest. On this basis, the GPMP2 [10] algorithm formulates the planning problem via a probabilistic graphical model and then converts the planning problem to a nonlinear least squares optimization problem.

For the motion planning problems with goal region, most of the previous work was based on sampling methods. In [11] [12], researchers sample multiple end-effector poses in the goal region and obtain the corresponding joint configurations by inverse kinematics (IK). Then these joint configurations are set as goals for a randomized planner. This approach has been

978-1-7281-1699-0/19/$31.00 ©2019 IEEE

confirmed to be neither probabilistically complete nor efficient. If the chosen goal configurations are unreachable, the planner will fail. To overcome this problem, RRT-JT algorithm [13] uses a gradient-descent heuristic based on the Jacobian-transpose to bias the tree toward a random workspace goal point. The random selection of workspace goal points ensures the probability completeness of the algorithm. In order to further improve the planning speed, IKBiRRT algorithm [14] uses a forward-searching tree rooted at the start configuration and a backward-searching tree which is seeded by sampling the workspace goal region with a given probability. By constructing a bidirectional search tree, the convergence speed of the algorithm is effectively improved.

The above methods are all based on the sampling algorithms. The final target configuration and the planned trajectory are feasible but not optimized. In order to quickly plan an optimized trajectory, we choose the GPMP2 [10] planner and improve it to solve the planning problems with goal region. For an irregular goal region in the three-dimensional workspace without posture constraints, we construct a distance field in the workspace. Then the closest distance and direction from any location of the workspace to the specified goal region can be easily obtained. Based on the distance information, we define a goal-region-constrained likelihood which specifies the probability that the position of end-effector is within the specific goal region. Then we view the likelihood as a posteriori and the problem is converted to a *maximum a posteriori* (MAP) problem. Finally, by numerical optimization, we plan an obstacle avoidance trajectory which moves the end-effector to an optimized goal configuration and the quality of the planned trajectory is improved effectively.

The rest of the paper is organized as follows. We define the problems we need to solve and explain how to use Gaussian processes to describe continuous-time trajectories in section II. Then the detailed algorithm description is presented in section III, including the construction of goal region distance fields and the derivation of trajectory optimization. Section IV presents the experimental results and illustrates the effectiveness and superiority of our algorithm. Finally several directions to improve the current work have been mentioned in the concluding section.

II. Preliminary Material

A. Problem Definition

For a n_q -dimensional kinematic redundant manipulator, the position t_p of the end-effector is related to the configuration coordinate q and can be compute by the kinematic map:

$$t_{\mathrm{p}} = f_{\mathrm{p}}(q) \tag{1}$$

Consider a specific goal region $\boldsymbol{G}$, which is a set of end-effector positions in the three-dimensional workspace without posture information, the goal-constrained motion planning problem is to plan a trajectory $q(s)$ in configuration space, $s \in [0,1]$, such that:

1. $t_{\mathrm{p}}(1) = f_{\mathrm{p}}(q(1)) \in \boldsymbol{G}$;
2. the robot does not collide with obstacles or itself throughout the planned trajectory.

B. Gaussian Processes as Continuous-time Trajectory Representation

For a continuous-time trajectory, the state at time t is

$$\xi(t) = \left\{ \{ \xi^{dr}(t) \}_{r=1}^{R} \right\}_{d=1}^{D} \tag{2}$$

where D is the dimension of the configuration space, and R represents the number of variables in each configuration dimension.

A vector-valued Gaussian process (GP) is a collection of random variables and each variable has a Gaussian distribution. Then a trajectory is considered as a sample from a vector-valued Gaussian process, $\xi(t) \sim GP(\mu(t), K(t,t'))$, with mean $\mu(t)$ and covariance $K(t,t')$ generated by a linear time varying stochastic differential equation (LTV-SDE)[15]

$$\begin{aligned} \dot{\xi}(t) &= \mathrm{A}(t)\xi(t) + \mathrm{u}(t) + \mathrm{F}(t)\,\mathrm{w}(t) \\ \mathrm{w}(t) &\sim GP(0, Q_c \delta(t - t')) \end{aligned} \tag{3}$$

where $\mathrm{A}(t)$ and $\mathrm{F}(t)$ are system matrices, $\mathrm{u}(t)$ is a known system control input, $\mathrm{w}(t)$ is the white noise process which is a Gaussian process with zero mean. Q_c is the power spectral density matrix and $\delta(\cdot)$ is the Dirac delta function.

The solution to the LTV-SDE is

$$\xi(t) = \Phi(t,t_0)\xi(t_0) + \int_{t_0}^{t} \Phi(t,s)(\mathrm{u}(s) + F(s)\,\mathrm{w}(s))ds, \tag{4}$$

where $\Phi(t,s)$ is the state transition matrix that propagates the state from s to t. Then the mean and covariance of the GP can be generated as following:

$$\mu(t) = \mathrm{E}[\xi(t)] = \Phi(t,t_0)\mu_0 + \int_{t_0}^{t} \Phi(t,s)\,\mathrm{u}(s)ds \tag{5}$$

$$\begin{aligned} K(t,t') &= \mathrm{E}[(\xi(t) - \mu(t))(\xi(t') - \mu(t'))^{\mathrm{T}}] \\ &= \Phi(t,t_0)K_0\Phi(t',t_0)^{\mathrm{T}} + \int_{t_0}^{\min(t,t')} \Phi(t,s)\,\mathrm{F}(s)\,\mathrm{Q}_c\,\mathrm{F}(s)^{\mathrm{T}}\Phi(t',s)^{\mathrm{T}}ds \end{aligned} \tag{6}$$

where μ_0, K_0 are initial mean and covariance of the first state.

The GP prior distribution is defined in terms of the mean μ and covariance K:

$$P(\xi) \propto \exp\{-\frac{1}{2}\|\xi - \mu\|_K^2\} \tag{7}$$

The major benefit arising from the Markovian property of the LTV-SDE in (3) is the fact that the inverse kernel matrix K^{-1} is exactly sparse block tri-diagonal

$$K^{-1} = A^{\mathrm{T}}Q^{-1}A^{-1} \tag{8}$$

where

$$A^{-1} = \begin{bmatrix} 1 & 0 & \cdots & 0 & 0 \\ -\Phi(t_1,t_0) & 1 & \cdots & 0 & 0 \\ 0 & -\Phi(t_2,t_1) & \ddots & \vdots & \vdots \\ 0 & 0 & \ddots & 0 & 0 \\ \vdots & \vdots & \cdots & 1 & 0 \\ 0 & 0 & \cdots & -\Phi(t_N,t_{N-1}) & 1 \end{bmatrix} \tag{9}$$

and $Q^{-1} = diag(K_0^{-1}, Q_1^{-1}, \cdots Q_N^{-1})$ with Q_i is given by

$$Q_i = \int_{t_{i-1}}^{t_i} \Phi(t_i, s)\,\mathrm{F}(s)\,\mathrm{Q}_c\,\mathrm{F}(s)^{\mathrm{T}}\,\Phi(t_i, s)^{\mathrm{T}}\,ds. \qquad (10)$$

The trajectory is going from t_0 to t_N and the detailed derivation process is shown in [15]. The state $\xi(\tau)$ at any time $\tau \in (t_i, t_{i+1})$ on the continuous trajectory can be approximated by Laplace's method:

$$\bar{\xi}(\tau) = \mu(\tau) + K(\tau)K^{-1}(\xi - \mu) \qquad (11)$$

where $K(\tau) = [K(\tau, t_0) \cdots K(\tau, t_N)]$. Because the inverse kernel matrix K^{-1} is sparse block tri-diagonal, the state $\bar{\xi}(\tau)$ can be computed by a linear combination of the two adjacent state function values as following:

$$\bar{\xi}(\tau) = \mu(\tau) + \Lambda(\tau)(\bar{\xi}_i - \mu_i) + \Psi(\tau)(\bar{\xi}_{i+1} - \mu_{i+1}) \qquad (12)$$

$$\Lambda(\tau) = \Phi(\tau, t_i) - \mathrm{Q}_\tau\,\Phi(\tau, t_i)^{\mathrm{T}}\,\mathrm{Q}_{i+1}^{-1}\,\Phi(t_{i+1}, t_i) \qquad (13)$$

$$\Psi(\tau) = \mathrm{Q}_\tau\,\Phi(\tau, t_i)^{\mathrm{T}}\,Q_{i+1}^{-1} \qquad (14)$$

where Q_i is given in (10) .

The Sparseness of the inverse kernel matrix effectively reduces the computational complexity of GP interpolation. Then the state $\xi(t)$ at any time on a continuous-time trajectory can be described by a Gaussian process.

III. Algorithm Description

A. Distance Field and Gradient Information

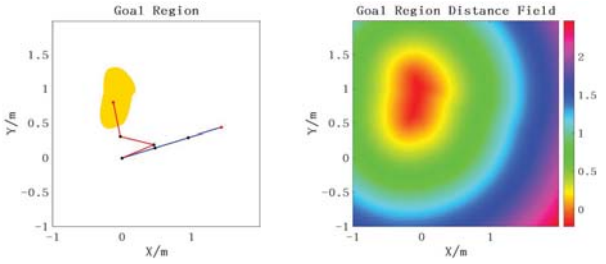

Fig. 2: The yellow region in the left image is the specific goal region and the closest distance from any intersections on the grid to the goal region is shown in the right figure.

The yellow region in the left image of Fig. 2 is the specific goal region and we rasterize the two-dimensional workspace first. Then using the fast distance field calculation method which is presented in [16] and has been publicly available, we compute the shortest Euclidean distance from each intersection on the grid to the goal region as shown in the right image. The distance field stores the distance information of all intersections.

Overly dense rasterization will exponentially increase the computational complexity and consumes a lot of storage space, so for limited rasterization, the goal position of the end-effector will inevitably fall into the grid rather than the intersections (e.g. the point T in Fig. 3). The shortest distance projection of a cell raster to the surface of the specific goal region is shown in Fig. 3. Considering that a cell raster is very small compared to the entire workspace and the projected portion is also very small compared to the entire goal region, we approximate the projected portion as linear. Then for any point in the grid, the closest distance can be estimated by bilinear interpolation or tri-linear interpolation method, corresponding to two-dimensional workspace and three-dimensional workspace respectively.

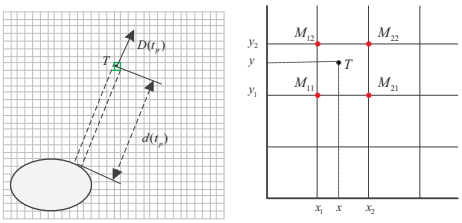

Fig. 3: the left image shows the shortest distance projection of a cell raster to the specific goal region and the detailed description for bilinear interpolation is shown in the right image.

In the right image of Fig. 3, points M_{11}, M_{12}, M_{21} and M_{22} are the four intersections on the grid. The closest distance d_{ij} of intersections is known and has been stored in the distance field. For a point T with coordinate $t_p = (x, y)$ in the grid, combined with the bilinear interpolation algorithm, the closest distance can be estimated:

$$\begin{aligned} d(t_p) \approx\ & (x_2 - x)(y_2 - y) * d_{11} + (x - x_1)(y_2 - y) * d_{21} \\ & + (x_2 - x)(y - y_1) * d_{12} + (x - x_1)(y - y_1) * d_{22} \end{aligned} \qquad (15)$$

where (x_i, y_j) is the coordinate of the intersection M_{ij}. In practice, the grids are unit-spaced and $x_2 - x_1 = y_2 - y_1 = 1$.

Then the gradient information of the distance for x and y can be obtained:

$$\begin{aligned} D(t_p) &\approx \left[\frac{\partial d(t_p)}{\partial x}\ \ \frac{\partial d(t_p)}{\partial y}\right]^{\mathrm{T}} \\ &= \begin{bmatrix} (y_2 - y)(d_{21} - d_{11}) + (y - y_1)(d_{22} - d_{12}) \\ (x_2 - x)(d_{12} - d_{11}) + (x - x_1)(d_{22} - d_{21}) \end{bmatrix} \end{aligned} \qquad (16)$$

For three-dimensional space, we use the tri-linear interpolation method to estimate the distance and the solution process is similar. For any position $t_p = (x, y, z)$ in three-dimensional workspace, the closest distance is:

$$\begin{aligned} d(t_p) \approx\ & (y_j - y)(z_k - z)\left((x_i - x)d_{i-1,j-1,k-1} + (x - x_{i-1})d_{i,j-1,k-1}\right) \\ & + (y - y_{j-1})(z_k - z)\left((x_i - x)d_{i-1,j,k-1} + (x - x_{i-1})d_{i,j,k-1}\right) \\ & + (y_j - y)(z - z_{k-1})\left((x_i - x)d_{i-1,j-1,k} + (x - x_{i-1})d_{i,j-1,k}\right) \\ & + (y - y_{j-1})(z - z_{k-1})\left((x_i - x)d_{i-1,j,k} + (x - x_{i-1})d_{i,j,k}\right) \end{aligned} \qquad (17)$$

where $d_{i,j,k}$ is the shortest Euclidean distance from the intersection point (x_i, y_j, z_k) to the goal region and

$$\begin{cases} x \in [x_i, x_{i+1}],\ y \in [y_j, y_{j+1}],\ z \in [z_k, z_{k+1}] \\ x_{i+1} - x_i = y_{j+1} - y_j = z_{k+1} - z_k = 1 \end{cases}. \qquad (18)$$

The gradient information is a $3 * 1$ vector by simply differentiating:

$$D(t_p) \approx \left[\frac{\partial d(t_p)}{\partial x} \ \frac{\partial d(t_p)}{\partial y} \ \frac{\partial d(t_p)}{\partial z} \right]^{\mathrm{T}}. \quad (19)$$

B. Goal-region-constrained Likelihood

Probabilistic inference provides an intuitive and efficient way to reason about the mapping between configurations and goal region. We define the likelihood that the position of the end-effector is within the specific goal region for a given goal configuration ξ_N and it is represented as:

$$L_{goal}(\xi_N \mid g) = P(g \mid \xi_N) \quad (20)$$

We define the likelihood as a distribution in the exponential family

$$L_{goal}(\xi_N \mid g) \propto \exp\{-\frac{1}{2}\left\| g(\xi_N) \right\|^2_{\Sigma_{goal}}\} \quad (21)$$

where $g(\xi_N)$ is a goal-region-constrained cost function and $\Sigma_{goal} = \sigma_{goal}$ is an arbitrary variance for this constraint, indicating how 'tight' the goal region constraint is.

In our implementation, we compute the cost function by

$$g(\xi_N) = \begin{cases} d(t_p(N)) + \varepsilon = d(f_p(\xi_N)) + \varepsilon & \text{if } d(t_p(N)) \geq -\varepsilon \\ 0 & \text{if } d(t_p(N)) < -\varepsilon \end{cases} \quad (22)$$

where ε is a safety distance to ensure that the end-effector is within the goal region. $t_p(N)$ is the goal position of end-effector and $f_p(\xi_N)$ is the forward kinematics that maps any goal configuration ξ_N to the workspace. $d(t_p(N))$ is the shortest distance from $t_p(N)$ to the specific goal region which has been computed in (15) and (17). It is a signed distance and the value will be negative when the end-effector is inside the goal region.

C. Trajectory Optimization

For a given Gaussian process prior $P(\xi)$, our purpose is to find an obstacle avoidance trajectory maximizing the likelihood in (21). The obstacle avoidance likelihood has been defined in [12] with exponential form:

$$L_{obs}(\xi_i \mid c_i = 0) = P(c_i = 0 \mid \xi_i) \propto \exp\{-\frac{1}{2}\left\| \mathrm{h}(\xi_i) \right\|^2_{\Sigma_{obs_i}}\} \quad (23)$$

where $h(\theta_i)$ is a vector-valued obstacle cost function and Σ_{obs_i} is a diagonal matrix and defined as, $\Sigma_{obs_i} = \sigma_{obs}\mathrm{I}$.

Then the planning problem can be converted to a MAP problem

$$\begin{aligned} \xi^* &= \underset{\xi}{\operatorname{argmax}} \left\{ P(\xi) P(g \mid \xi_N) \prod_i P(c_i \mid \xi_i) \right\} \\ &= \underset{\xi}{\operatorname{argmin}} \left\{ -\log\left(P(\xi) P(g \mid \xi_N) \prod_i P(c_i \mid \xi_i) \right) \right\} \\ &= \underset{\xi}{\operatorname{argmin}} \left\{ \frac{1}{2}\left\| \xi - \mu \right\|^2_K + \frac{1}{2}\left\| g(\xi_N) \right\|^2_{\Sigma_{goal}} + \frac{1}{2}\left\| h(\xi) \right\|^2_{\Sigma_{obs}} \right\} \end{aligned} \quad (24)$$

where $h(\xi) = [\mathrm{h}(\xi_0) \cdots \mathrm{h}(\xi_N)]^{\mathrm{T}}$.

After the above derivation, the terms in (24) can be viewed as 'cost' to be minimized. Then the problem has been converted to a nonlinear least squares optimization problem and many numerical tools are available. Linearizing the nonlinear obstacle cost function around the current trajectory ξ as is motioned in [12]

$$h(\xi + \delta\xi) \approx h(\xi) + \mathrm{H}\,\delta\xi, \ \mathrm{H} = \frac{dh}{d\xi}\Big|_{\xi} \quad (25)$$

For the nonlinear goal-region-constrained cost function $g(\xi_N)$, the linearized form around ξ_N can be expressed as:

$$g(\xi_N + \delta\xi_N) \approx g(\xi_N) + \mathrm{G}\,\delta\xi_N. \quad (26)$$

Combined with (22), when $d(t_p(N)) > -\varepsilon$, we can get

$$\mathrm{G} = \frac{\partial\left(g(\xi_N)\right)}{\partial\xi_N} = \frac{\partial d(f_p(\xi_N))}{\partial f_p(\xi_N)} \frac{\partial f_p(\xi_N)}{\partial \xi_N} \quad (27)$$

$$\frac{\partial d(f_p(\xi_N))}{\partial f_p(\xi_N)} = \frac{\partial d(t_p(N))}{\partial t_p(N)} = D(t_p(N))^{\mathrm{T}} \quad (28)$$

$$\frac{\partial f_p(\xi_N)}{\partial \xi_N} = J_p(\xi_N) \quad (29)$$

where $D(t_p(N))^{\mathrm{T}}$ is a $1 * n_t$ vector ($n_t = 2 \text{ or } 3$) which represents the gradient form of the shortest distance in n_t -dimensional workspace and we have derived it in (16) and (19). $J_p(\xi_N)$ is a $n_t * n_q$ position Jacobian matrix of the n_q -joint manipulator. So the gradient information for the goal-region-constrained cost is

$$\mathrm{G} = \frac{\partial\left(g(\xi_N)\right)}{\partial\xi_N} = \begin{cases} D(t_p(N))^{\mathrm{T}} * J_p(\xi_N) & \text{if } d(t_p(N)) > -\varepsilon \\ \mathbf{0.5} & \text{if } d(t_p(N)) = -\varepsilon \\ \mathbf{0} & \text{if } d(t_p(N)) < -\varepsilon \end{cases} \quad (30)$$

Finally, Equation (24) is converted to a linear least squares problem

$$\begin{aligned} \delta\xi^* = \underset{\delta\xi}{\operatorname{argmin}} \Big\{ & \frac{1}{2}\left\| \xi + \delta\xi - \mu \right\|^2_K + \frac{1}{2}\left\| g(\xi_N) + \mathrm{G}\,\delta\xi_N \right\|^2_{\Sigma_{goal}} \\ & + \frac{1}{2}\left\| h(\xi) + \mathrm{H}\,\delta\xi \right\|^2_{\Sigma_{obs}} \Big\}. \end{aligned} \quad (31)$$

VI. Simulation Experiments

In our implementation, we use the 'constant-velocity' GP prior [15] with white noise injected in acceleration, $\ddot{\theta} = w(t)$. $\ddot{\theta}$ is a n_q -dimensional vector which represents the accelerations of each joint. Then the trajectory is generated by the LTV-SDE in (3) with

$$\xi(t) = \begin{bmatrix} \theta \\ \dot{\theta} \end{bmatrix}, \mathrm{A}(t) = \begin{bmatrix} \mathbf{0} & \mathbf{1} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}, \mathrm{u}(t) = \mathbf{0}, \mathrm{F}(\mathrm{t}) = \begin{bmatrix} \mathbf{0} \\ \mathbf{1} \end{bmatrix}. \quad (32)$$

In this case, given $\Delta t_i = t_i - t_{i-1}$, we have

$$\Phi(\mathrm{t,s}) = \begin{bmatrix} \mathbf{1} & (t-s)\mathbf{1} \\ \mathbf{0} & \mathbf{1} \end{bmatrix}, \ Q_i = \begin{bmatrix} \Delta t_i^3 Q_c / 3 & \Delta t_i^2 Q_c / 2 \\ \Delta t_i^2 Q_c / 2 & \Delta t_i Q_c \end{bmatrix}. \quad (33)$$

Then the inverse kernel matrix K^{-1} in (8) is obtained and the GP prior cost in (24) can be derived as

$$\frac{1}{2}\|\xi-\mu\|_K^2=\frac{1}{2}(\xi-\mu)^{\mathrm{T}}K^{-1}(\xi-\mu)$$
$$=\frac{1}{2}\sum_{i=1}^{N}\left\|\begin{bmatrix}\theta_{i-1}+\Delta t_i\cdot\dot{\theta}_{i-1}-\theta_i\\ \dot{\theta}_{i-1}-\dot{\theta}_i\end{bmatrix}\right\|_{Q_i}^2. \quad (34)$$

So applying such prior will minimize the actuator acceleration in configuration space and Q_c has an effect on smoothness.

In our experiments, we realize our algorithm based on the GPMP2 C++ library [10] and solve the nonlinear least squares optimization problem with the GTSAM C++ library [17]. We use the Levenberg-Marquardt algorithm (initial λ = 0.01; termination condition: maximum 100 iterations, or relative error smaller than 10^{-4}) in GTSAM to solve the optimization problem. The comparison experiments based on the sampling method is carried out on an open source motion planning platform MoveIt. All Experiments are performed on Ubuntu system with a 1.9-GHz Intel Core i5-4300U processor and 4 GB of RAM.

A. *Motion Planning for 3-DOF Planar Manipulator*

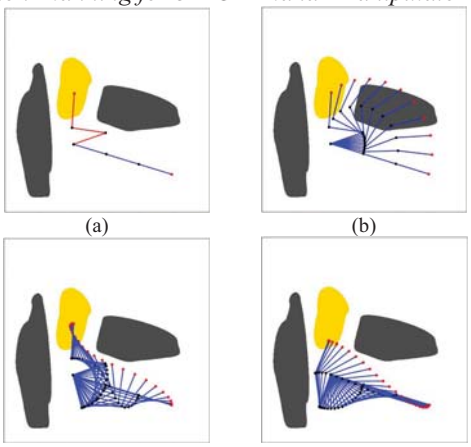

Fig. 4: The image (a) shows the given start and goal configurations and image (b) shows the initial constant-velocity straight line trajectory. The optimized trajectories with single goal and goal region are shown in image (c) and (d).

The first scenario we describe is to plan an obstacle-avoidance trajectory that moves the 3-DOF manipulator from the initial configuration to the yellow goal region in Fig. 4 (a). The algorithm is initialized by a constant-velocity straight line trajectory in configuration space as shown in Fig. 4 (b). Comparing the optimized trajectories in Fig. 4 (c) and (d), our algorithm can obtain a better goal configuration apparently through an optimized approach as shown in Fig. 4 (d).

In order to prove the superiority of our proposed algorithm, we randomly selected 23 goal positions in the goal region and obtained the corresponding valid goal configurations through Inverse Kinematics (IK) solution. The smoothness cost was set to $Q_c=1$ and the obstacle cost was set to $\sigma_{obs}=0.1$. We gave a strong constraint for the goal region with $\sigma_{goal}=0.05$ and the safety distance was set to $\varepsilon=0.1$. Then, we ran both the single goal and goal region algorithms based on GPMP2. The comparative experimental performance data is shown in Table I and all data is averaged. The trajectory cost includes the GP prior cost and obstacle cost in (25). It embodies the smoothness of the planning trajectory and the ability to avoid obstacles.

TABLE I
THE COMPARATIVE EXPERIMENTAL PERFORMANCE DATA (AVERAGED)

	Success Rate (%)	Trajectory Cost	Iterative Times	Planning Time (s)
Single Goal	91.3	4.819	21.5	**0.0329**
Goal Region	**100**	**1.707**	34.9	0.0558

From the data in Table I we can surmise that the choice of goal configuration heavily influences the quality of the trajectory planned by GPMP2. Compared to the single-goal algorithm, the trajectory generated by our algorithm is significantly improved in quality and the cost has been reduced by 64.6% on average. Meanwhile, we experimented with the 23 randomly sampled target configurations. The single-goal algorithm failed twice and all of the trajectories generated by our algorithm are collision free and eventually fall into the goal region. So overall, the success rate will also increase. However, due to the need to optimize the goal configuration in goal region, our algorithm requires more iterative times when the initial given goal configuration is farther away from the final optimized goal configuration. Therefore, the planning time will increase compared to the original algorithm, but the difference is small and it is within tens of milliseconds.

B. *Motion Planning for 7-DOF WAM Manipulator*

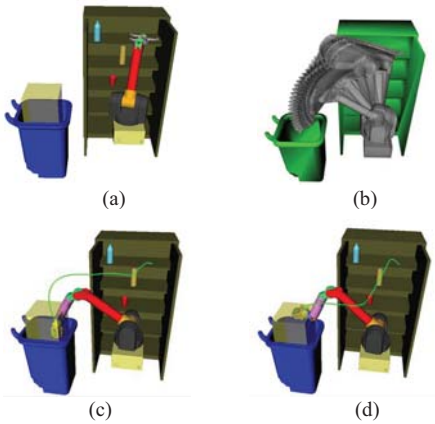

Fig. 5: The image (a) shows the initial state of the manipulator and the goal region is the yellow transparent area. The image (b) shows the trajectory generated by IKBiRRT. The end-effector trajectory planned by GPMP2 with single goal and goal region are the green lines in (c) and (d).

We also carried out simulation experiments on the 7-DOF WAW manipulator and the task is to throw the objects on the cabinet into the recycling bin as shown in Fig. 5 (a). In actual applications, by laser or depth camera, the outline and the position information of the recycling bin can be obtained and then determine the goal region. Here the recycling bin model is

built in the STL format and we obtained its outline information through the slicing technique. Then the goal region is simplified into a cylindrical area as shown in the yellow transparent area of Fig. 5 (a). Here we set the smoothness cost $Q_c = 1$ and obstacle cost $\sigma_{obs} = 0.02$. Then we still gave a strong constraint for the goal region with $\sigma_{goal} = 0.005$. For the single goal and goal region algorithms based on GPMP2, we randomly sampled 19 valid goal configurations and the initialized trajectory is a straight line with constant velocity in configuration space. Meanwhile, we constructed the same simulation environment on MoveIt as shown in Fig. 5 (b) and performed 50 experiments using the IKBiRRT algorithm [14] for the same task. The parameter P_{sample} in IKBiRRT is set to 0.3. The planning time of IKBiRRT includes two parts: path planning and post-processing.

TABLE II
THE PERFORMANCE DATA FOR WAM MANIPULATOR

	IKBiRRT	GPMP2_Single Goal	GPMP2_Goal Region
Trajectory Cost	-	1428.35	**1004.94**
Avg Time (s)	1.304	**0.143**	0.163
Max Time (s)	2.312	0.281	**0.264**

As shown in Table II, the planning time of GPMP2 is much lower than IKBiRRT and the planning speed has increased by 87.5%. Compared to setting single goal configuration in GPMP2, our proposed algorithm is slightly increased in planning time, but the cost of the planned trajectory has been significantly reduced by 30%. In general, the trajectory optimization approach is far superior to sampling-based approach in planning speed. For the planning problems with goal region, different from IKBiRRT which randomly samples in the goal region, we take the goal region constraint as an optimization objective. Then the planner can find an optimized goal configuration and plan a smoother obstacle avoidance trajectory.

V. CONCLUSION

Motion planning problems with workspace goal region are very common in manipulator applications. Different from the original algorithms all of which are based on RRT planner, we use the trajectory optimization method and make improvements on the basis of GPMP2. Especially for irregular goal region, we construct a goal region distance field in workspace and derive the closest distance from the end effector to the goal region and its differential form. Then we take the goal region constraint as an optimization goal and define a goal-region-constrained likelihood which specifies the probability that the position of end-effector is within goal region. Combined with the original GP prior and obstacle avoidance likelihood, we convert the planning problem into a nonlinear least squares optimization problem and use the Levenberg-Marquardt algorithm to solve it. We performed multiple simulation experiments on a 3-DOF planar robot and a 7-DOF WAM manipulator respectively. The experimental results show that our proposed algorithm is far superior to the original sampling-based method in planning speed. Meanwhile, compared to randomly given a valid goal configuration within goal region, our planner can find an optimized goal configuration and plan a smoother obstacle avoidance trajectory. The future work includes implementation of the proposed algorithm to a physical industrial manipulator and determination of the goal region by laser or depth camera.

REFERENCES

[1] 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.

[2] A. Stentz, "Optimal and efficient path planning for partially-known environments", *IEEE International Conference on Robotics and Automation*, vol. 4, pp. 3310-3317, 1994.

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

[4] S. Lavalle, "Rapidly-exploring random trees: a new tool for path planning", *Research Report*, pp. 293-308, 1998.

[5] S. Karaman and E. Frazzoli, "Incremental Sampling-based Algorithms for Optimal Motion Planning", *Proceedings of Robotics: Science and Systems*, pp. 5326-5332, 2010.

[6] J. Nasir, et al, "RRT*-SMART: A Rapid Convergence Implementation of RRT*", *International Journal of Advanced Robotic Systems*, vol. 10, no. 7, 2013.

[7] N. Ratliff, M. Zucker, J. A. Bagnell and S. Srinivasa, "Chomp: Gradient optimization techniques for efficient motion planning", *IEEE International Conference on Robotics and Automation*, pp. 489-494, 2009.

[8] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor and S. Schaal, "Stomp: Stochastic trajectory optimization for motion planning", *IEEE International Conference on Robotics and Automation*, pp. 4569-4574, 2011.

[9] M. Mukadam, X. Yan and B. Boots, "Gaussian Process Motion planning", *IEEE International Conference on Robotics and Automation*, pp. 9-15, 2016.

[10] J. Dong, M. Mukadam, F. Dellaert, and B. Boots, "Motion Planning as Probabilistic Inference using Gaussian Processes and Factor Graphs", *Proceedings of Robotics: Science and Systems*, 2016.

[11] M. Stilman, J.U. Schamburek, J. Kuffner, and T. Asfour, "Manipulation Planning Among Movable Obstacles", *International Conference on Intelligent Robots and Systems*, pp. 3327–3332, 2007.

[12] Y. Hirano, K. Kitahama, and S. Yoshizawa, "Image-based object recognition and dexterous hand/arm motion planning using rrts for grasping in cluttered scene", *International Conference on Intelligent Robots and Systems*, pp. 2041–2046, 2005.

[13] J. M. Vandeweghe, D. Ferguson and S. Srinivasa, "Randomized path planning for redundant manipulators without inverse kinematics", *International Conference on Humanoid Robots*, pp. 477–482, 2007.

[14] D. Berenson, et al, "Manipulation planning with Workspace Goal Regions", *IEEE International Conference on Robotics and Automation*, pp. 618- 624, 2009.

[15] T. Barfoot, C. H. Tong and S. Sarkka, "Batch continuous-time trajectory estimation as exactly sparse Gaussian process regression", *Proceedings of Robotics: Science and Systems*, pp. 221-238, 2014.

[16] Y. Mishchenko, "A fast algorithm for computation of discrete Euclidean distance transform in three or more dimensions on vector processing architectures", *Signal, Image and Video Processing*, vol. 9, no. 1, pp. 19-27, 2015.

[17] F. Dellaert, "Factor Graphs and GTSAM: A Hands-on Introduction", *Georgia Institute of Technology*, 2012.