Multi-agent Gaussian Process Motion Planning via Probabilistic Inference ⋆

Luka Petrović, Ivan Marković, Marija Seder *

* *University of Zagreb Faculty of Electrical Engineering and Computing, Laboratory for Autonomous Systems and Mobile Robotics, Croatia (e-mail: {luka.petrovic, ivan.markovic, marija.seder}@fer.hr)*

Abstract: This paper deals with motion planning for multiple agents by representing the problem as a simultaneous optimization of every agent's trajectory. Each trajectory is considered as a sample from a one-dimensional continuous-time Gaussian process (GP) generated by a linear time-varying stochastic differential equation driven by white noise. By formulating the planning problem as probabilistic inference on a factor graph, the structure of the pertaining GP can be exploited to find the solution efficiently using numerical optimization. In contrast to planning each agent's trajectory individually, where only the current poses of other agents are taken into account, we propose simultaneous planning of multiple trajectories that works in a predictive manner. It takes into account the information about each agent's whereabouts at every future time instant, since full trajectories of each agent are found jointly during a single optimization procedure. We compare the proposed method to an individual trajectory planning approach, demonstrating significant improvement in both success rate and computational efficiency.

Keywords: motion planning, trajectory optimization, multi-agent system, probabilistic inference, factor graphs

1. INTRODUCTION

Motion planning is an indispensable skill for robots that aspire to navigate through an environment without collisions. Motion planning algorithms attempt to generate trajectories through the robot's configuration space that are both feasible and optimal based on some performance criterion that may vary depending on the task, robot, or environment. Motion planning algorithms that can be executed in real time are highly encouraged, mostly because they allow for fast replanning in response to environment changes.

A significant amount of recent work has focused on trajectory optimization and related problems. Trajectory optimization methods start with an initial trajectory and then minimize an objective function in order to optimize the trajectory. Ratliff et al. (2009) and Zucker et al. (2013), in their work abbreviated as CHOMP, proposed utilizing a precomputed signed distance field for fast collision checking and using covariant gradient descent to minimize obstacle and smoothness costs. Kalakrishnan et al. (2011) developed a stochastic trajectory optimization method (STOMP) that samples a series of noisy trajectories to explore the space around an initial trajectory which are then combined to produce an updated trajectory with lower cost. The key trait of STOMP is its ability to optimize non-differentiable constraints. An important shortcoming of CHOMP and STOMP is the need for many trajectory states in order to reason about fine resolution obstacle representations or find feasible solutions when there are many constraints. Schulman et al. (2013), Schulman et al. (2014), in their framwork called TrajOpt, formulate motion planning as sequential quadratic programming. The key feature of TrajOpt is the ability to solve complex motion planning problems with few states since swept volumes are considered to ensure continuous-time safety. However, if the smoothness is required in the output trajectory, either a densely parametrized trajectory or post-processing of the trajectory might still be needed thus increasing computation time. In order to overcome the computational cost incurred by using large number of states, the Gaussian process motion planning family of algorithms (Mukadam et al. (2016), Dong et al. (2016), Huang et al. (2017)) use continous-time trajectory representation. Mukadam et al. (2016) parametrize the trajectory with a few support states and then use Gaussian process (GP) interpolation to query the trajectory at any time of interest. Dong et al. (2016), in their framework called GPMP2, represent continuous-time trajectories as samples from a GP and then formulate the planning problem as probabilistic inference, generating fast solutions by exploiting the sparsity of the underlying linear system. A useful property of GPMP2 is its extensibility and applicability for wide range of problems (Rana et al. (2017), Mukadam et al. (2017a), Marić et al. (2018)) and in this paper we also rely on the aforementioned framework. All mentioned methods are primarily designed for planning motion of one robot in static environments. In many challenging problems, such as warehouse management, delivery and construction, a single robot may not be able to achieve desired tasks and therefore multi-robot teams are required. Multi-robot teams are also more robust to malfunctions, since one

⋆ This research has been supported by the European Regional Development Fund under the grant KK.01.1.1.01.0009 (DATACROSS).

robot can take over the tasks of another robot in case of failure.

In this paper, we propose using a continous time Gaussian process trajectory representations in order to plan motion for every robot in a multi-agent system. We augment the method proposed in Dong et al. (2016) and consider multi-agent trajectory optimization from a probabilistic inference perspective, optimizing all of the trajectories concurrently. By optimizing the trajectories at the same time, our method takes into account the information where each agent will be at every future time instant, thus working in a predictive manner. We evaluated our approach in simulation and compared it to planning each agent's trajectory indvidually.

The rest of the paper is organized as follows. Section 2 presents Gaussian processes as trajectory representations. In Section 3, the method for simultaneous multi-agent trajectory optimization as probabilistic inference is proposed. Section 4 describes implementation aspects of the proposed method. Section 5 demonstrates the main results in simulation and Section 6 concludes the paper.

2. GAUSSIAN PROCESSES AS TRAJECTORY REPRESENTATIONS

A continuous-time trajectory is considered as a sample from a vector-valued Gaussian process (GP), $x(t) \sim \mathcal{GP}(\mu(t), K(t,t'))$, with mean $\mu(t)$ and covariance $K(t,t')$, generated by a linear time-varying stochastic differential equation (LTV-SDE)

$$\dot{x}(t) = A(t)x(t) + u(t) + F(t)w(t) \tag{1}$$

where A, F are system matrices, u is a known control input and $w(t)$ is generated by a white noise process. The white noise process is itself a GP with zero mean value

$$w(t) \sim \mathcal{GP}(0, Q_c\delta(t-t')), \tag{2}$$

where Q_c is a power spectral density matrix .

The solution of the LTV-SDE in (1) is generated by the mean and covariance of the GP:

$$\mu(t) = \Phi(t,t_0)\mu_0 + \int_{t_0}^{t} \Phi(t,s)u(s)ds \tag{3}$$

$$K(t,t') = \Phi(t,t_0)K_0\Phi(t',t_0)^T + \int_{t_0}^{min(t,t')} \Phi(t,s)F(s)Q_cF(s)^T\Phi(t',s)ds, \tag{4}$$

where μ_0, K_0 are initial mean and covariance of the first state, and $\Phi(t,s)$ is the state transition matrix (Barfoot et al. (2014)).

The GP prior distribution is then given in terms of its mean μ and covariance K:

$$p(x) \propto \exp\{-\frac{1}{2}\|x-\mu\|_K^2\}. \tag{5}$$

One major benefit of using Gaussian processes to model continuous-time trajectory in motion planning is the possibility to query the planned state $x(\tau)$ at any time of interest τ, and not only at discrete time instants. If the prior proposed in (5) is used, GP interpolation can be performed efficiently due to the Markovian property of the LTV-SDE given in (1). State $x(\tau)$ at $\tau \in [t_i, t_{i+1}]$ is a function only of its neighboring states (Dong et al. (2016))

$$x(\tau) = \mu(\tau) + \Lambda(\tau)(x_i - \mu_i) + \Psi(\tau)(x_{i+1} - \mu_{i+1}), \tag{6}$$

$$\Lambda(\tau) = \Phi(\tau, t_i) - \Psi(\tau)\Phi(t_{i+1}, t_i), \tag{7}$$

$$\Psi(\tau) = Q_{i,\tau}\Phi(t_{i+1},\tau)^T Q_{i,i+1}^{-1}, \tag{8}$$

where

$$Q_{a,b} = \int_{t_a}^{t_b} \Phi(b,s)F(s)Q_cF(s)^T\Phi(b,s)^T ds. \tag{9}$$

The fact that any state $x(\tau)$ can be computed in $\mathcal{O}(1)$ complexity can be exploited for efficient computation of obstacle avoidance costs, as explained in Section 3.

Another major benefit arising from the aforementioned Markovian property of the LTV-SDE in (1) is the fact that the inverse kernel matrix K^{-1} of this prior is exactly sparse block tridiagonal (Barfoot et al. (2014))

$$K^{-1} = A^{-T}Q^{-1}A^{-1} \tag{10}$$

where

$$A^{-1} = \begin{bmatrix} 1 & 0 & ... & 0 & 0 \\ -\Phi(t_1,t_0) & 1 & ... & 0 & 0 \\ 0 & -\Phi(t_2,t_1) & \ddots & \vdots & \vdots \\ \vdots & \vdots & \ddots & 1 & 0 \\ 0 & 0 & ... & -\Phi(t_N,t_{N-1}) & 1 \end{bmatrix} \tag{11}$$

and

$$Q^{-1} = \text{diag}(K_0^{-1}, Q_{0,1}^{-1}, ..., Q_{N-1,N}^{-1}) \tag{12}$$

with $Q_{a,b}$ given in (9), the trajectory going from t_0 to t_N, and K_0 being the initial covariance. As it will be shown in Section 3, this kernel allows for fast, structure-exploiting inference.

3. MULTI-AGENT TRAJECTORY OPTIMIZATION AS PROBABILISTIC INFERENCE

In this section we formulate the multi-agent trajectory optimization problem as probabilistic inference. The presented formulation is predicated on the work of Dong et al. (2016) and it represents the extension of their trajectory optimization method to multi-robot systems.

To formulate the trajectory optimization problem as probabilistic inference, we seek to find a trajectory parametrized by x given desired events e. The posterior density of x given events e can be computed via Bayes' rule from a prior and likelihood

$$p(x|e) = p(x)p(e|x)/p(e) \propto p(x)p(e|x) \tag{13}$$

where $p(x)$ represents the prior on x which encourages smoothness of the trajectory, while $p(e|x)$ represents the probability of the desired events occuring given x. The optimal trajectory x is found by maximizing the posterior $p(x|e)$, using the *maximum a posteriori* (MAP) estimator

$$x^* = \arg\max_x p(x)p(e|x), \tag{14}$$

where

$$p(e|x) = \prod_i p(e|x_i) \tag{15}$$

with x_i being the configuration at discrete time instant t_i. The conditional distributions $p(e|x_i)$ specify the likelihood of desired events occuring at the configuration x_i

$$L(x_i|e) \propto p(e|x_i). \tag{16}$$

In motion planning context, desired event is avoidance of collisions and therefore conditional distribution $p(e|x_i)$ specifies the likelihood that the configuration x_i is collision free

$$L_{obs}(x_i|c_i = 0) \propto p(c_i = 0|x_i). \tag{17}$$

In our case, for each agent there are two possible types of collision; collision with a static obstacle in the environment and collision with another agent. Since the aforementioned types of collision are independent events, the likelihood of trajectory x being free of collisions can therefore be considered as the product of two likelihoods:

$$L_{obs}(x|c = 0) = L_{stat}(x|c_{obs} = 0)L_{mul}(x|c_{mul} = 0), \tag{18}$$

where the first the term specifies the probability of being clear of collisions with static obstacles and the second term specifies the probability of being clear of collisions with other agents.

For each agent j, the likelihood of being free of collision with static obstacles is defined as a distribution of the exponential family (Dong et al., 2016):

$$L_{stat}(x_j|c_{obs} = 0) \propto \exp\{-\frac{1}{2}\big\|h(x_j)\big\|^2_{\Sigma_{obs}}\}, \tag{19}$$

where $h(x)$ is a vector-valued obstacle cost function and Σ_{obs} a diagonal matrix and the hyperparameter of the distribution. In the same manner, we define the likelihood of being collision-free with other agents as a distribution of the exponential family that is a product of probabilities of being free of collision with every other agent

$$L_{mul}(x_j|c_{mul} = 0) \propto \prod_{\substack{j'=1 \\ j'\neq j}}^{n_{ag}} \exp\{-\frac{1}{2}\big\|g(x_j, x_{j'})\big\|^2_{\Sigma_{mul}}\}, \tag{20}$$

where $g(x_j, x_{j'})$ is a vector-valued function that defines the cost of two agents j and j' being close to each other, n_{ag} is number of agents and Σ_{mul} is a hyperparameter of the distribution.

Combining Eq. (19) and Eq. (20), the total likelihood of agent j being free of collision is obtained

$$L_{obs}(x_j|c = 0) \propto \exp\{-\frac{1}{2}\big\|h(x_j)\big\|^2_{\Sigma_{obs}}\} \prod_{\substack{j'=1 \\ j'\neq j}}^{n_{ag}} \exp\{-\frac{1}{2}\big\|g(x_j, x_{j'})\big\|^2_{\Sigma_{mul}}\}. \tag{21}$$

Deriving the MAP trajectory from Eq. (14), Eq. (5) and Eq. (21) in a similar manner to Dong et al. (2016), our *maximum a posteriori* trajectory for each agent is

$$x^*_j = \arg\min_{x_j} \{\frac{1}{2}\big\|x_j - \mu_j\big\|^2_{K_j} + \frac{1}{2}\big\|h(x_j)\big\|^2_{\Sigma_{obs}} + \frac{1}{2}\sum_{\substack{j'=1 \\ j'\neq j}}^{n_{ag}} \big\|g(x_j, x_{j'})\big\|^2_{\Sigma_{mul}}\}. \tag{22}$$

This is a nonlinear least squares problem that can be solved with iterative approaches, such as Gauss-Newton or Levenberg-Marquardt.

To obtain the MAP trajectory by solving the aforementioned optimization problem with iterative approaches, we need a linearized approximation of (22). Converting the nonlinear least squares problem to a linear problem around operating point $\bar{x}_j$, the following expression for the optimal perturbation δx^*_j is obtained

$$\delta x^*_j = \arg\min_{\delta x_j}\Big\{\frac{1}{2}\big\|\bar{x}_j + \delta x_j - \mu_j\big\|^2_{K_j} + \frac{1}{2}\big\|h(\bar{x}_j) + H_j\delta x_j\big\|^2_{\Sigma_{obs}} + \frac{1}{2}\sum_{\substack{j'=1 \\ j'\neq j}}^{n_{ag}} \big\|g(\bar{x}_j, x_{j'}) + G_j\delta x_j\big\|^2_{\Sigma_{mul}}\Big\}, \tag{23}$$

where H_j is the Jacobian matrix of $h(x_j)$

$$H_j = \frac{\partial h}{\partial x_j}\bigg|_{\bar{x}_j}, \tag{24}$$

and G_j is the partial derivative of $g(x_j, x_{j'})$

$$G_j = \frac{\partial g}{\partial x_j}\bigg|_{(\bar{x}_j, x_{j'})}. \tag{25}$$

The optimal perturbation δx^*_j is obtained by solving the following linear system

$$\Big(K_j^{-1} + H_j^T\Sigma_{obs}^{-1}H_j + \sum_{\substack{j'=1 \\ j'\neq j}}^{n_{ag}} G_j^T\Sigma_{mul}^{-1}G_j\Big)\delta x^*_j = K_j^{-1}(\mu_j - \bar{x}_j) - H_j^T\Sigma_{obs}^{-1}h(\bar{x}_j) - \sum_{\substack{j'=1 \\ j'\neq j}}^{n_{ag}} G_j^T\Sigma_{mul}^{-1}g(\bar{x}_j, x_{j'}). \tag{26}$$

This MAP trajectory optimization can be represented as inference on a factor graph (Kschischang et al. (2001)). The fact that the system in (26) is linear and sparse can be exploited for finding the solution efficiently. In our case, the posterior distribution given in Eq. (13) can be factorized similarly to Mukadam et al. (2017b):

$$P(x|e) \propto \prod_{t_i}\prod_{c_j} f^{gp}_j(x^j_i, x^j_{i+1}) f^{obs}_{i,j}(x^j_i) \prod_{c_{j'}} f^{mul}_{i,j,j'}(x^j_i, x^{j'}_i) \prod_{\tau=1}^{n_p} f^{intp}_{i,j,j',\tau}(x^j_i, x^j_{i+1}, x^{j'}_i, x^{j'}_{i+1}), \tag{27}$$

where f^{gp} represents factor corresponding to a GP prior, f^{obs} represents the cost of collision with static obstacles, f^{mul} represents the cost of collision with other agents and f^{intp} represents collision cost calculated for n_p interpolated states that can be obtained by using (6). The factor graph defined in (27) is depicted in Fig. 3 for a simple case of trajectory optimization problem with two agents and three states.

4. IMPLEMENTATION DETAILS

4.1 GP prior

In our implementation, for dynamics of our robot we use the double integrator linear system with white noise injected in acceleration, meaning that the trajectory is generated by the LTV-SDE in (1) with

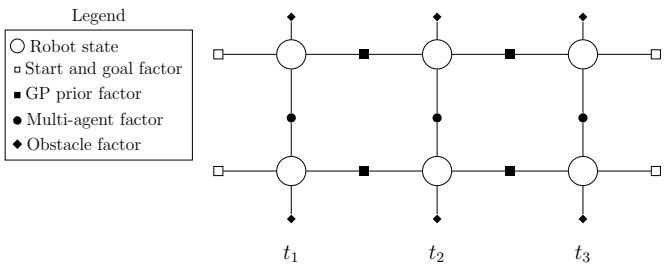

Fig. 1. A simple illustration of the factor graph describing an example multi-robot trajectory optimization problem. GP interpolation factors present between states are omitted for clarity.

$$A = \begin{bmatrix} 0 & \mathrm{I} \\ 0 & 0 \end{bmatrix},\ u(t) = 0,\ F(t) = \begin{bmatrix} 0 \\ \mathrm{I} \end{bmatrix}. \tag{28}$$

This represents a constant velocity GP prior which is centered around a zero acceleration trajectory (Barfoot et al. (2014)). Applying such a prior will minimize actuator velocity in the configuration space, thus minimizing the energy consumption and giving the physical meaning of smoothness. Choosing the noise covariance Q_c has an effect on smoothness, with smaller values penalizing deviation from the prior more.

4.2 Avoidance of collision with static obstacles

In (19) we defined the likelihood of one agent j being free of collision with static obstacles which relies upon a vector-valued obstacle cost function $h(x_j)$. In our case, for the obstacle cost $h(x_j)$ we use the hinge loss function with a precomputed signed distance field similarly to Dong et al. (2016) and Mukadam et al. (2017b). The pertaining hinge loss function is defined as:

$$h(x_j) = \begin{cases} \varepsilon_{obs} - d_{s,j} & \text{if } d_{s,j} \leq \varepsilon_{obs} \\ 0 & \text{if } d_{s,j} > \varepsilon_{obs} \end{cases}, \tag{29}$$

where $d_{s,j}$ is the signed distance, and ε_{obs} is the *safety distance* indicating the boundary of the *danger area* near obstacle surfaces (Mukadam et al. (2017b)).

4.3 Avoidance of collision with other agents

In (20) we defined the likelihood of one agent j being free of collision with other agent j'. This likelihood relies on a vector-valued function $g(x_j, x_{j'})$, which we defined as the hinge loss function:

$$g(x_j, x_{j'}) = \begin{cases} \varepsilon_{mul} - d_{j,j'} & \text{if } d_{j,j'} \leq \varepsilon_{mul} \\ 0 & \text{if } d_{j,j'} > \varepsilon_{mul} \end{cases}, \tag{30}$$

where $d_{j,j'}$ denotes $d(x_j, x_{j'})$, the Euclidean distance between two agents j, and j' and ε_{mul} is a *safety distance* indicating the boundary of the area inside which we anticipate a collision. If the parameter ε_{mul} is set to zero, multi-agent obstacle cost would always be zero and our planner would work exactly the same as GPMP2 (Dong et al. (2016)) for each agent. Higher values of ε_{mul} enable robots to anticipate each other's future trajectories and to adapt them accordingly in the next iteration of optimization, giving our algorithm a predictive property.

To obtain the MAP trajectory in (23), the proposed method requires G_j, the partial derivative of the hinge loss function defined in (31). This partial derivative used in our implementation can be analytically obtained:

$$\frac{\partial g(x_j, x_{j'})}{\partial x_j} = \begin{cases} \dfrac{x_{j'} - x_j}{d(x_j, x_{j'})}. & \text{if } d_{j,j'} < \varepsilon_{mul} \\ 0.5 & \text{if } d_{j,j'} = \varepsilon_{mul} \\ 0 & \text{if } d_{j,j'} > \varepsilon_{mul} \end{cases}. \tag{31}$$

Parameters Σ_{obs} and Σ_{mul}, needed to fully implement static and multi-agent obstacle likelihoods in (19) and (20) are defined by isotropic diagonal matrices $\Sigma_{obs} = \sigma_{obs}^2 I$ and $\Sigma_{mul} = \sigma_{mul}^2 I$, where σ_{obs} and σ_{mul} represent *obstacle cost weights*. Reducing the value of σ_{obs} causes the optimization to place more weight on avoiding collision with static obstacles, while reducing the value of σ_{mul} causes the optimization to place more weight on avoiding collision with other agents.

4.4 Software implementation

In our experiments we use the GPMP2 C++ library (Dong et al. (2016), Dong et al. (2017)), and its respective MATLAB toolbox, which is based on the GTSAM C++ library (Dellaert (2012), Dellaert and Kaess (2006)). Experiments are performed on a system with a 3.8-GHz Intel Core i7-7700HQ processor and 16 GB of RAM.

5. EXPERIMENTAL RESULTS

5.1 Formation control

In this section we present the results of our method in a quantitative manner. We used the proposed approach to switch the positions of 2D holonomic circular robots with radius $r = 1\,\mathrm{m}$ inside a formation in an obstacle free simulation environment. We did this for every possible permutation of positions inside a formation for cases of three, four and five agents and compared the success rate and computation time of our approach to the GPMP2 framework (Dong et al. (2016)) where each agent's trajectory is planned individually at every time step. In three and five robot simulation the formation is a triangle, while in four robot simulation it's a square. That means that we tested and compared algorithms on the total of 150 unique planning problems. When using the GPMP2 framework, for every agent the others were incorporated in its signed distance field (SDF), meaning that at every time step the SDF had to be changed and the trajectory had to be replanned.

All trajectories were initialized as a constant-velocity straight line trajectory in configuration space. Total time of execution for every case is set as $t_{total} = 10\,\mathrm{s}$ and all trajectories were parametrized with 10 equidistant support states such that 9 points are interpolated between any two states (91 states effectively). The parameters used for GPMP2 are $\varepsilon_{obs} = 2$, $\Sigma_{obs} = 0.3$, while the parameters used for our approach are $\varepsilon_{mul} = 15$, $\Sigma_{mul} = 0.7$. It makes sense to use relatively small ε_{obs} in comparison to ε_{mul} since in GPMP2 the agents only react to each other locally, and in our approach it is desireable that every agent knows each other's position at all times in order to calculate

Table 1. Comparison of success rates (percentages) for our method and replanning with GPMP2 for every possible permutation of positions inside formations of 3, 4 and 5 agents

Number of robots	GPMP2	MUL-GPMP
3	100	100
4	87.5	100
5	70.8	100

Table 2. Comparison of average execution times (ms) for our method and replanning with GPMP2 for every possible permutation of positions inside formations of 3, 4 and 5 agents

Number of robots	GPMP2	MUL-GPMP
3	196	22
4	264	37
5	457	59

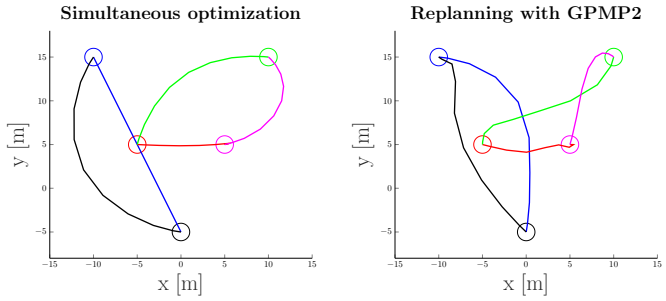

Fig. 2. Example of switching formation with the proposed simultaenous optimization and with replanning with GPMP2. The proposed algorithm produces visibly smoother trajectories due to its predictive nature.

the trajectories that avoid collisions. The success rates comparison is showed in Table 1 and the computational times comparison is showed in Table 2.

Since computing the SDF and replanning the trajectory is relatively computationally demanding in comparison to computing the *multi-agent* factor in (20), it is not surprising that our approach achieves significantly faster computational times. Incorporating current positions of other agents inside an agent's SDF means that the GPMP2 framework, unlike the proposed method, can only react to the changes in the environment and not anticipate them. Thus the better success rates of our approach were also expected. Our method was able to successfully solve all of the given 150 motion planning problems, while the GPMP2 failed in total 38 cases. Due to the predictive nature of the proposed approach, the generated trajectories are also visibly smoother which is demonstrated in Fig. 2 for one example of five robot formation change. Note that for the case of replanning with GPMP2 better results could be achieved by parameter tuning, possibly using a grid search, but that is relatively computationally demanding and possibly unattainable in the real world circumstances.

5.2 Complex static environment

This experiment shows qualitatively the capability of our approach to generate trajectories that can ensure successful simultaneous motion of two agents in complex static environments. Our goal was to switch positions of two planar holonomic robots that start in two rooms which are separated by a narrow hallway. The radius of each robot is $r = 1\,\text{m}$, and the width of the hallway is $w = 3.6\,\text{m}$, which is smaller than two robots diameters combined, meaning that robots cannot pass each other in the hallway. Since the environment is complex, if the initial trajectory of each robot is set as a straight line, the optimization gets stuck in the local minimum and the collisions are not avoided. To solve that problem, we set the initial trajectories as the paths obtained by A^* algorithm with constant velocities. When computing the path with the A^* algorithm, we downsampled the grid representing the environment so that the initial path was obtained in acceptable time. Since the paths obtained by the A^* algorithm are only used as priors in our method, possible loss of information about small sized obstacles when downsampling is not concerning.

The described setup represents a challenging task for existing motion planning algorithms, and most of them would result in redundant motion; robots would move towards each other and meet in the middle of the hallway, where one of them would have to turn back to make way for the other. In our case, however, due to the fact that the proposed approach works in a predictive manner, after exiting the room one robot turns away from its goal and waits for the other one to enter its destination room before proceeding to travel to its goal. The downside of the proposed method is the number of the optimization hyperparameters that need to be set, for example static and multi-agent obstacle cost factor covariances Σ_{obs} and Σ_{mul}. In this specific example, the optimization hyperparameters Σ_{obs}, Σ_{mul}, ε_{obs}, ε_{mul} were tuned via exhaustive grid search. The described environment and the result of the conducted experiment are shown in Fig. 3.

6. CONCLUSION AND FUTURE WORK

In this paper we have presented a fast trajectory optimization method for multi-agent motion planning. We considered each trajectory as a sample from a continuous time Gaussian process generated by linear time-varying stochastic differential equation driven by white noise. We formulated the multi-agent planning problem as probabilistic inference on a factor graph, and thus were able to exploit the structure of the mentioned GP to find the solution efficiently using numerical optimization. The proposed approach works in a predictive manner since each agent's trajectory is optimized simultaneously. We tested our approach in simulation and compared it to planning each trajectory indvidually, demonstrating significant improvement in both success rate and computational efficiency.

In future work, it would be interesting to investigate how different priors affect the result of optimization. Another potentially interesting direction would be to explore the possibility of planning robot's trajectory in a dynamic environment. If the trajectory of a dynamic obstacle could be predicted, the avoidance of such obstacle would be achieved using the same concepts introduced in the proposed approach.

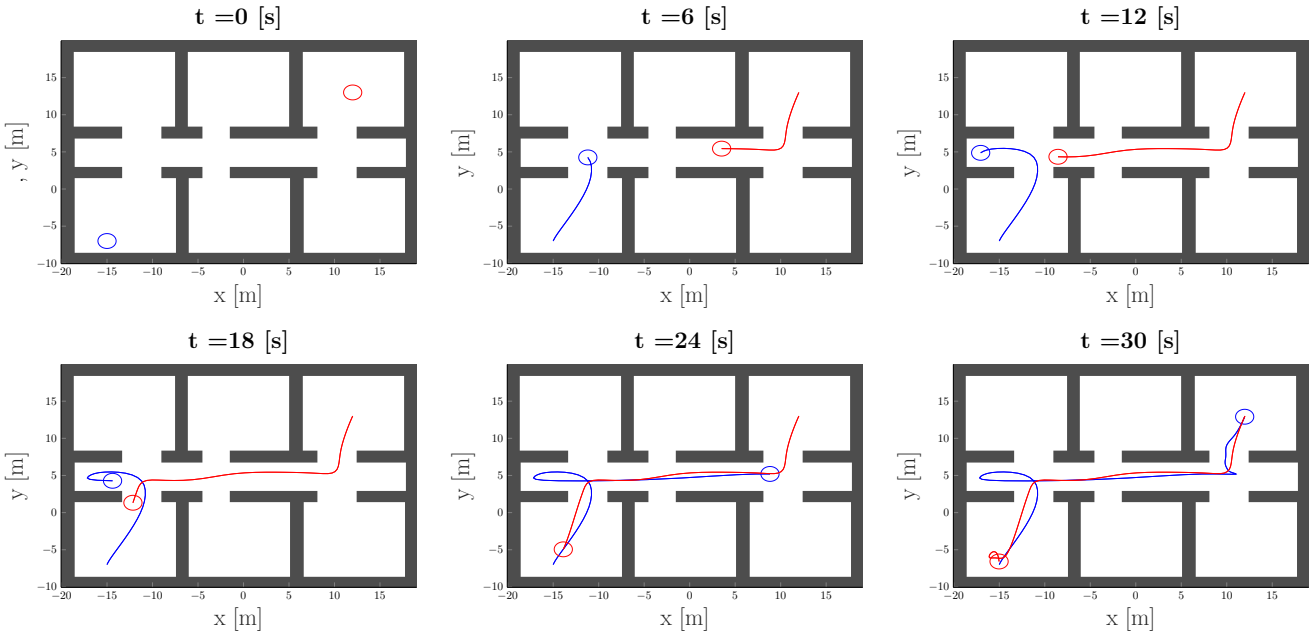

Fig. 3. The example of trajectories planned with our approach for two agents in a complex static environment consisting of rooms with narrow passage between them. Robots anticipate each other's trajectories and one robot makes way for the other one.

REFERENCES

Barfoot, T.D., Tong, C.H., and Särkkä, S. (2014). Batch continuous-time trajectory estimation as exactly sparse gaussian process regression. In *Robotics: Science and Systems*.

Dellaert, F. (2012). Factor graphs and gtsam: A hands-on introduction. Technical report, Georgia Institute of Technology.

Dellaert, F. and Kaess, M. (2006). Square root sam: Simultaneous localization and mapping via square root information smoothing. *The International Journal of Robotics Research*, 25(12), 1181–1203.

Dong, J., Boots, B., and Dellaert, F. (2017). Sparse gaussian processes for continuous-time trajectory estimation on matrix lie groups. *arXiv preprint arXiv:1705.06020*.

Dong, J., Mukadam, M., Dellaert, F., and Boots, B. (2016). Motion planning as probabilistic inference using gaussian processes and factor graphs. In *Proceedings of Robotics: Science and Systems (RSS-2016)*.

Huang, E., Mukadam, M., Liu, Z., and Boots, B. (2017). Motion planning with graph-based trajectories and gaussian process inference. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, 5591–5598.

Kalakrishnan, M., Chitta, S., Theodorou, E., Pastor, P., and Schaal, S. (2011). Stomp: Stochastic trajectory optimization for motion planning. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, 4569–4574.

Kschischang, F.R., Frey, B.J., and Loeliger, H.A. (2001). Factor graphs and the sum-product algorithm. *IEEE Transactions on information theory*, 47(2), 498–519.

Marić, F., Limoyo, O., Petrović, L., Petrović, I., and Kelly, J. (2018). Singularity avoidance as manipulability maximization using continuous time gaussian processes. *arXiv preprint arXiv:1803.09493*.

Mukadam, M., Dong, J., Dellaert, F., and Boots, B. (2017a). Simultaneous trajectory estimation and planning via probabilistic inference. In *Proceedings of Robotics: Science and Systems (RSS)*.

Mukadam, M., Dong, J., Yan, X., Dellaert, F., and Boots, B. (2017b). Continuous-time gaussian process motion planning via probabilistic inference. *arXiv preprint arXiv:1707.07383*.

Mukadam, M., Yan, X., and Boots, B. (2016). Gaussian process motion planning. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, 9–15.

Rana, M.A., Mukadam, M., Ahmadzadeh, S.R., Chernova, S., and Boots, B. (2017). Towards robust skill generalization: Unifying learning from demonstration and motion planning. In *Conference on Robot Learning*, 109–118.

Ratliff, N., Zucker, M., Bagnell, J.A., and Srinivasa, S. (2009). Chomp: Gradient optimization techniques for efficient motion planning. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, 489–494.

Schulman, J., Duan, Y., Ho, J., Lee, A., Awwal, I., Bradlow, H., Pan, J., Patil, S., Goldberg, K., and Abbeel, P. (2014). Motion planning with sequential convex optimization and convex collision checking. *The International Journal of Robotics Research*, 33(9), 1251–1270.

Schulman, J., Ho, J., Lee, A.X., Awwal, I., Bradlow, H., and Abbeel, P. (2013). Finding locally optimal, collision-free trajectories with sequential convex optimization. In *Robotics: science and systems*, volume 9, 1–10.

Zucker, M., Ratliff, N., Dragan, A.D., Pivtoraiko, M., Klingensmith, M., Dellin, C.M., Bagnell, J.A., and Srinivasa, S.S. (2013). Chomp: Covariant hamiltonian optimization for motion planning. *The International Journal of Robotics Research*, 32(9-10), 1164–1193.