RUMOR-ROBUST DECENTRALIZED GAUSSIAN PROCESS (GP) LEARNING, FUSION, AND PLANNING FOR SENSOR NETWORK ON MULTIPLE MOVING TARGETS TRACKING

A Thesis

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Master of Science

by

Zhihao Liao

May 2020

© 2020 Zhihao Liao

ALL RIGHTS RESERVED

ABSTRACT

A decentralized GP learning, fusion, and planning (RESIN) algorithm for a mobile sensor network to actively learn the motion pattern of multiple moving targets, and thus planning for each sensor to pursue the targets based on the information entropy was proposed. RESIN is combined with a decentralized GP fusion method which is robust to rumor propagation and computational efficient by using the weighted exponential product based on Chernoff information, and an information-driven path planning (IPP) method that is able to generate the most information sensitive path for the mobile sensor network by using sequential planning and fusing each sensor with its predecessors' planning information. Various numerical simulations were done to show that RESIN is effective and could achieve near-optimal performance for the sensor network. Also, RESIN shows more applicability while in the situation that the number of sensors is less than number of targets.

BIOGRAPHICAL SKETCH

Zhihao Liao is currently an M.S. student in the Laboratory for Intelligent Systems and Controls (LISC) at Cornell University. He received his B.S. degree in Mechanical Engineering from Tsinghua University. His research interests include robotics, machine learning, computer vision, and optimal path planning.

This document is dedicated to all Cornell graduate students.

ACKNOWLEDGEMENTS

I would like to give my sincere thank to Prof. Silvia Ferrari for her guidance and advising in my research during my Master program at Cornell University. Her kindness and patience encouraged me to go through all the problems. Also, I want to thank Prof. Bharath Hariharan for his suggestion on my thesis and defense. Dr. Chang Liu worked with me together to build the theoretical part of this thesis and Dr. Jake Gemerek and Qingze Huo provided lots of help in the research into Scene Flow. Last bu not least, I would like to thank all other LISC members, Pingping Zhu, Taylor Clawson, Julian Morelli, Shi Chang, Jaejeong Shin, Yucheng Chen, Hengye Yang, Junyi Dong, Xinyu Gao, Dongheng Jing, Yifeng Shi, Shenghao Liu for their assistance and help in the lab.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION AND MOTIVATION

Sensor planning problem has been increasing attracting attention and is widely used for a various of applications such as environmental monitoring [29, 36], video surveillance [21, 32, 11, 12] and the internet of things [14]. From these applications, it can be learned that during the practical situation, usually the region of interest (ROI) are greatly larger than the area of the sensor's field of view (FOV). As a result, learning the target's behavior and planning for the sensor's motion becomes a crucial point in order to improve the performance of collecting the information of the targets in the ROI.

One example of such research is the Pursuit–Evasion Games [30], the problem of a network of sensors pursuing a team of evaders (targets) while concurrently building a map in an unknown environment. This problem was defined as a probabilistic game theoretical framework with a finite 2-D environment containing an unknown number of fixed obstacles. Evaders and pursuers are placed in the environment and are allowed to move to the unoccupied space. In this research, the motion of the evader is modeled as a Markov process and the planning is based on the policy to maximized the probability of sensors capturing and containing an evader. Moreover, in the work of [10], the energy consumed of the sensor pursuing the evader is also taken into consideration and the pursuit policy are thus further optimized. Then, in [15], a new modeling method of target's motion is proposed, which deployed a camera installed on a helicopter for monitoring the traffic of the Boston area. The traffic motion is then modeled using Dirichlet Process Gaussian Processes (DPGP), which are proved to be faster and more appropriate for the decentralization of the state space than markov process.

In this thesis, a “rumor-robust” decentralized GP learning and fusion approach which aims to prevent the rumor propagation in GP fusion is proposed. The approach is demonstrated on a collaborative sensor network to perform decentralized target modeling and trajectory prediction. Also, a decentralized information-driven path planning (IPP) approach is also presented in this thesis for controlling and coordinating sensor trajectories such that the sensors could obtain the most informative target measurements under the communication constraints.

In the following content of the thesis, Chapter 4 would discuss about the optical flow method for the motion computation of the targets. The decentralized GP learning and fusion approach would be described in Chapter 5 and the decentralized path planning algorithm would be presented in chapter 6. In chapter 7, numerical simulations would be done to show the accuracy and efficiency of RESIN algorithm proposed in this thesis.

CHAPTER 2

BACKGROUND

Nonparametric models, such as gaussian process, has been showing its efficiency and flexibility in modeling multi-targets. For a multivariable gaussian distribution $f \sim GP(\mu, k)$, in order to model the probability distribution $P(f_* \mid x, D)$, we would use a GP prior $P(f \mid x) \sim N(\mu, \Sigma)$ and condition it based on the training data D, to make prediction by maximise the log-marginal likelihood [8]:

$$\theta =_\theta log(P(f \mid x, D)) = -\frac{1}{2}(f^T(K + \sigma_\epsilon^2 I)^{-1} f + log|K + \sigma_\epsilon^2 I|) \tag{2.1}$$

Define $K = k(S, S) \in^{N \times N}$ as the kernel matrix. With the trained hyperparameter θ, the posterior distribution could be predicted by:

$$\begin{aligned} E[f_*] &= k_*^T (K + \sigma_\epsilon^2 I)^{-1} f \\ var[f_*] &= k_{**} - k^T (K + \sigma_\epsilon^2 I)^{-1} k_* \end{aligned} \tag{2.2}$$

Where f_* is the predicted motion of the target, s_* is the test input, $k_* = k(S, s_*)$ and $k_{**} = k(s_*, s_*)$.

For a single gaussian process, while dealing with multi-targets trajectory learning, although it performs robust to unaligned, noisy measurements and provides a flexible representation for each individual motion pattern, it is not able to model different target with different destinations and motion patterns. In order to solve this problem, we need to mixture the gaussian models of different motion patterns[15]. On way is to use Dirichlet Process to mixture Gaussian

Process.

The Dirichlet Process (DP) is a distribution over discrete distributions, usually the total number of distributions is unbounded, which means there could be nearly infinite motion patterns and the total probability would be still modeled as 1. Among them, a few patterns are expected to be followed by the target for the majority of time. For these motion patterns, the prior probability that motion pattern z_i of target trajectory t^i belongs to an existing motion pattern b_j is modeled as:

$$P(z_i = j \mid z_{-i}, \alpha) = \frac{n_j}{N - 1 + \alpha} \tag{2.3}$$

Where z_{-i} denotes the motion pattern assignments for the remaining trajectories, α represents the concentration parameter of the Dirichlet process, n_j is the number of trajectories assigned to motion pattern b_j , and N is the total number of observed trajectories. And the probability that motion pattern z_i is a new motion pattern is:

$$P(z_i = M + 1 \mid z_{-i}, \alpha) = \frac{\alpha}{N - 1 + \alpha} \tag{2.4}$$

Where M is the number of observed motion patterns. For any finite measurable motion pattern $B_{i_{i=1}}^{n}$, the following holds:

$$[P(B_1)...P(B_n)]^T \sim Dir(\alpha H(B_1), ..., \alpha H(B_n)) \tag{2.5}$$

Where "Dir" denotes the Ditichlet distribution. Now we can do the DPGP mixture:

$$\begin{aligned} \theta_i, \pi &\sim DP(\alpha, GP_0), i = 1, ..., \infty \\ G_j &\sim Cat(\pi), j = 1, ..., N \\ f_{G_j}(x) &\sim GP(\theta_{Gj}, \Psi), x \in W, j = 1, ..., N \end{aligned} \tag{2.6}$$

Where "GP" is gaussian process, $GP_0 = GP(0, \Psi)$, "Cat(π)" denotes an M-dimensional categorical distribution with probability mass function π. With the DPGP mixture we are now able to model the dynamic targets from sensor measurements that are obtained over a period of time $[t_o, t_f]$.

Compared to a single, non-stationary GP, DPGP has the following advantages: (a) It preserves the use of the well-studied and widely-applied stationary covariance functions, many of which exhibit the locality property and are computationally friendly with only a few (unknown) hyperparameters to be trained, (b) the required number of locally stationary GPs can automatically grow with the increasing complexity of the phenomenon, and (c) each locally stationary GP only incurs cubic time in the size of the observations that are local to its corresponding area of prediction instead of over the entire phenomenon.[27]

Decentralized GP learning and fusion approaches have been developed by distributing the computation into independent local agents and each agent just operates on a subset of the whole data. There are two representative classes for decentralized GP learning and fusion, the mixture of experts (MoEs) [35] and the product of experts (PoEs) [6, 8]. In MoEs, each agent would locally learn its own GP model for different partition of the whole state space and then make a global prediction by synthesizing all the local predictions together. Each local prediction would be assigned a weight based on the agent's domain corresponding to the training data. In contrast, the agents in PoEs would share the same state space with each agent independently learn a GP model just using a subset of the training data. The global prediction is therefore made using Bayes rule and the independence assumption of local predictions. Comparing with MoEs, the PoEs method is more efficient in training and thus attracts great interest. However, current PoE approaches are not able to be directly applied in data fusion within a sensor network, since the PoE-based approaches would suffer the problem of rumor propagation, which means the common information between local agents, i.e. the simultaneous measurements of the same target, would be redundantly used and therefore may lead into incorrect fusion results [5].

Information-driven planning refers to the problem of determining the best control policy for the sensor to gather the measurement over a future period of time [9]. One of the previous methods for decentralized IPP is a decentralized, gradient-based control approach which assumes all-to-all agent communications [13] where each sensor's measurement and the gradient of the objective function are communicated constantly with the whole network, and the trajectories are proved to be converged into a Nash equilibrium. However since multiple iterations are required for the convergence of gradient-based optimization, this method incurs large communication burden. Another one is a distributed planner proposed to learn spatial-temporal processes using GPs [33], where each sensor constantly communicates with its neighboring sensors and computes the optimal trajectories for both itself and its neighbors to achieve coordination. While the planning is fully decentralized, the exchange of measurements and planning for neighbors also result in high communication burden and redundant computation. More recently, a multi-robot online sensing strategy for using GP to construct the communication maps was proposed [19].

This work uses a leader-follower paradigm, where each pair of leader-follower sensors is manually defined and the leader generates plans to coordinate with its follower. However, no coordination is ensured between different pairs.

So to overcome these challenges, this thesis proposes a rumor-robust decentralized GP learning, fusion, and planning (RESIN[1]) approach for a mobile sensor network to actively learn target motion models. To deal with time-varying target motion models, a spatio-temporal kernel is used in GP modeling. A rumor-robust decentralized PoE-based GP learning and fusion algorithm is then applied to combine individual sensors' local prediction of target trajectories into a globally consistent one. The GP learning and fusion approach is computationally efficient and is able to avoid rumor propagation in the sensor network.

[1]RESIN is the acronym of "Rumor-robust decentralized gp lEarning, fuSIon, and planNing".

CHAPTER 3

PROBLEM FORMULATION

This thesis considers the situation that multiple targets $\mathcal{T}_1, ..., \mathcal{T}_m$, are moving with different motion patterns in a Euclidean workspace $\mathcal{W} \subset \mathbb{R}^2$. A network of N mobile robots, where there is a fixed sensor installed on each robot, are deployed to learn the motion models of the targets. For each robot, the sensor consists of a fixed stereo camera, which can extract the motion of the target it observed in the form of scene flow and a wireless communication device that enables information transfer between sensors. For certain target $\mathcal{T}_i, i \in \{1, ...m\}$, its trajectory could be represented as a differentiable continuous function $\mathbf{f}_i : \mathbb{R}^2 \times \mathbb{R} \rightarrow \mathbb{R}^2$ defined over the workspace $\mathcal{W}$. The function maps the position of the target to its velocity, as the following shows,

$$\dot{\mathbf{x}}_i(t) = \mathbf{f}_i\left[\mathbf{x}_i(t), t\right] \triangleq \mathbf{v}_i(t), \tag{3.1}$$

where $\mathbf{x}_i(t) \in \mathcal{W}$ and $\mathbf{v}_i(t) \in \mathbb{R}^2$ respectively represent the position and velocity of the target's center of mass.

The sensor's state is defined by a $\mathbb{R}^4$ matrix $\mathbf{s} = [s_x \quad s_y \quad s_\theta \quad s_v]^T$, where $\mathbf{s} = [s_x \quad s_y]^T$ denotes the position of the sensor in the workspace $\mathcal{W}$, $s_\theta \in [0, 2\pi)$ denotes the orientation of the sensor and $s_v > 0$ denotes the linear velocity of the robot. The control input to the robot is defined as a $\mathbb{R}^2$ matrix $\mathbf{u} = [\omega \quad a]^T$, where a represents the linear acceleration and ω represents the angular velocity. With the time interval be defined as $\Delta T > 0$, the kinematic model of the robot j could be represented by the following difference equation which denotes the state transformation between the kth step and the $(k + 1)$th step,

$$\mathbf{s}_j(k+\Delta T) = \mathbf{s}_j(k) + \begin{bmatrix} s_v(k)\cos(s_\theta(k)) \\ s_v(k)\sin(s_\theta(k)) \\ \mathbf{u}_j(k) \end{bmatrix} \Delta T \tag{3.2}$$

For simplification, in the following content of this thesis, the $\mathbb{R}^4 \times \mathbb{R}^2 \to \mathbb{R}^4$ function (3.2) will be represented in the form of,

$$\mathbf{s}_j(k+\Delta T) = \mathbf{g}(\mathbf{s}_j(k), \mathbf{u}_j(k), \Delta T). \tag{3.3}$$

Each sensor could measure the position and velocity of the target within its field of view (FOV) by computing the sparse scene flow of the target [18]. This thesis assumes the FOV of the sensor is defined as $\mathcal{F}\left(\mathbf{s}_j(k)\right) = \left\{\mathbf{w} \in \mathcal{W} \mid \left\|[s_x(k) \quad s_y(k)]^T - \mathbf{w}\right\|_2 \leq r_j\right\}$, where $r_j > 0$ denotes the measuring range of the jth sensor. The measurement model of the camera could then be defined as the following function,

$$\mathbf{z}_{ij}(k) = \begin{cases} \mathbf{v}_i(k) + \boldsymbol{\varepsilon} & \text{if } \mathbf{x}_i(k) \in \mathcal{F}\left(\mathbf{s}_j(k)\right) \\ \emptyset & \text{if } \mathbf{x}_i(k) \notin \mathcal{F}\left(\mathbf{s}_j(k)\right) \end{cases}, \tag{3.4}$$

where $\boldsymbol{\varepsilon} \in \mathbb{R}^2$ is an additive zero-mean Gaussian white noise added to the measurement, with the distribution $\boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_\varepsilon)$, where $\boldsymbol{\Sigma}_\varepsilon = \epsilon_0^2\mathbf{I}$, and $\mathbf{z}_{ij}(k) \in \mathbb{R}^2$ denotes the velocity of the ith target measured by the jth sensor. For simplicity, this thesis assumes that the target could be accurately measured by the sensor provided that it's in the FOV of the sensor.

A communication network will then be built by connecting all the sensors. Let a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ represent this communication network, where

$\mathcal{V} = \{1, \ldots, N\}$ is the set of sensor indices and $\mathcal{E} = \left\{e_{jl} \in \{0,1\} \mid j, l = 1, \ldots, N\right\}$ denotes the edge set. Sensors j and l can exchange information when there is a communication link between them, i.e. $e_{ij} = 1$ [24].

In practical applications, usually the communication range between sensors is limited, so in this thesis, sensors are divided into groups which is formed by neighboring sensors. It's assumed that sensors within the same group can communicate with each other at each time step.

For the sensors to keep learning the models of the target's motion, an accurate prediction on the target's future positions is required. Based on the prediction, the sensor could then actively planning the trajectory it is supposed to move along with, so that the sensor is able to keep the target within its FOV and extract the measurement from the target continuously.

The approaches for solving the two problems introduced above are going to be discussed in the following sections.

3.1 Decentralized Learning and Fusion

In this thesis, an approach using Gaussian Process for predicting the target motion and thus computing the future trajectories of the target is deployed. In practical applications, since the communication between the sensors is limited. To conduct centralized GP learning would be difficult as the data transfer are only happened within neighboring robots. So decentralized GP serves as a better choices. For each sensor, the GP model runs locally and the local prediction would then be fused into a global one as the final prediction, which, as is fused by the observation and prediction of every sensor in the sensor network, could help reduce the uncertainty of the estimation of target state. One typical issue in decentralized fusion is the rumor propagation, where the common information in sensors' local data are double counted. To better explain this, let $P_j\left(\mathbf{x}(k) \mid \mathbf{Z}_j(k)\right)$ and $P_l\left(\mathbf{x}(k) \mid \mathbf{Z}_l(k)\right)$ represent the probability density function (pdf) of local estimates of $\mathbf{x}(k)$ from sensors j and l, respectively. Since in current PoEs methods, $P_j\left(\mathbf{x}(k) \mid \mathbf{Z}_j(k)\right)$ and $P_l\left(\mathbf{x}(k) \mid \mathbf{Z}_l(k)\right)$ are assumed independent, as a result the fusion would just be the product of them, which is $P\left(\mathbf{x}(k) \mid \mathbf{Z}_j(k)\right) P\left(\mathbf{x}(k) \mid \mathbf{Z}_l(k)\right)$. If a target is measured both by sensors j and l, then simply multiplying the two pdfs would result in incorrect global prediction, which is called rumor propagation. The way to avoid rumor propagation during the estimation of target's state is to ensure that the common information could be tracked and removed during computation. In other words, the fused pdf should be,

$$P\left(\mathbf{x}(k) \mid \mathbf{Z}_j(k) \cup \mathbf{Z}_l(k)\right) \propto \frac{P\left(\mathbf{x}(k) \mid \mathbf{Z}_j(k)\right) P\left(\mathbf{x}(k) \mid \mathbf{Z}_l(k)\right)}{P\left(\mathbf{x}(k) \mid \mathbf{Z}_j(k) \cap \mathbf{Z}_l(k)\right)}, \tag{3.5}$$

where the denominator is the conditional probability distribution based on common information between sensors j and l [1]. However, the cost to conduct such a computation is usually expensive. So in this thesis, an approach of decentralized GP fusion was proposed, which could combine the local predictions and thus generate a global consistent prediction on the target motion while avoiding rumor propagation.

3.2 Information-driven Path Planning (IPP) Algorithm

The second problem is the trajectory planning for the robot so that the sensor is able to keep the targets in its FOV. In this thesis the planning algorithm is formulated as an optimal control problem. Suppose $\boldsymbol{u}_j\left(k:k_f\right) = \left[\boldsymbol{u}_j^T(k) \quad \dots \quad \boldsymbol{u}_j^T\left(k_f\right)\right]^T$ as a sequence of control input on the jth sensor over the planning interval $[k, k_f]$, and the set $\boldsymbol{U}\left(k:k_f\right) = \left[\boldsymbol{u}_1\left(k:k_f\right) \quad \dots \quad \boldsymbol{u}_N\left(k:k_f\right)\right]$ denote the control inputs of all N sensors, which could be computed by maximizing a cost function $J\left(\boldsymbol{U}\left(k:k_f\right)\right)$ under the system constrains. The problem is then formulated as a optimization problem, as the following equation represents,

$$
\begin{aligned}
\boldsymbol{U}^*\left(k:k_f\right) &= \arg\max_{\boldsymbol{U}\left(k:k_f\right)} J\left(\boldsymbol{U}\left(k:k_f\right)\right) \\
s.t.\ \ \mathbf{s}_j(t+\Delta T) &= \mathbf{g}\left(\mathbf{s}_j(t), \mathbf{u}_i(t), \Delta T\right) \\
\mathbf{s}_j(t) &\in \mathcal{S},\ \mathbf{u}_j(t) \in \mathcal{U} \\
t &= k, \dots, k_f - 1,\ j = 1, \dots, N,
\end{aligned}
\tag{3.6}
$$

where $\mathcal{S}$ and $\mathcal{U}$ respectively represent the sensor state and control input.

In this thesis, mutual information (MI) is deployed in the cost function, since it has been proved effective in Information-driven path planning [13, 31, 32]. Define $\mathbf{X}\left(k:k_f\right) = \left[\mathbf{x}_1\left(k:k_f\right) \quad \dots \quad \mathbf{x}_m\left(k:k_f\right)\right]$ as the prediction of target positions during the planning procedure, which could be obtained by decentralized GP fusion. Meanwhile, based on the predicted trajectories of the targets, the trajectories of the sensors could be predicted given the control input into the robots. Then with the prediction of both targets and sensors, the sensor measurement could therefore be predicted . Define the predicted measurement of all the sensors as $\mathbf{Z}\left(k:k_f\right)$, then the MI between $\mathbf{X}\left(k:k_f\right)$ and $\mathbf{Z}\left(k:k_f\right)$ can be defined as $J\left(\boldsymbol{U}\left(k : k_f\right)\right)$, given the existing measurements stored in the sensor, i.e.,

$$J\left(\boldsymbol{U}\left(k : k_f\right)\right) = I\left(\mathbf{X}\left(k : k_f\right); \mathbf{Z}\left(k : k_f\right) \mid \mathbf{Z}\left(1 : k-1\right)\right). \tag{3.7}$$

Although in function (3.7), $\boldsymbol{U}\left(k : k_f\right)$ does not explicitly appear, the predicted sensor positions $\mathbf{S}\left(k : k_f\right)$ could be computed since it directly depends on the control input into the robot, then it would determine the expected sensor measurements $\mathbf{Z}\left(k : k_f\right)$ and thus, affecting the cost function.

Generally, it's computationally expensive to solve the centralized IPP as the search space would grow exponentially with respect to the growing of sensor number and the planning horizon. So this thesis proposes a decentralized IPP algorithm for distributing the computation to different sensors and thus improve the efficiency through parallel computation.

CHAPTER 4

SCENE FLOW EXTRACTION AND TARGET VELOCITY COMPUTATION

One way to measure the motion of the object using visual algorithm is to extract the scene flow from the object. Scene flow is an approximation of the motion of each volume voxel of the image taken by the camera, which could be computed under the 3D Motion Constraint Equation [3]. Define a 3D volume voxel $I(x, y, z, t)$, where x, y, z denotes the spatial position of the voxel and t denotes the time. As the voxel is moving continuously, after a small time interval, the same voxel now at time $t+\delta t$ could be represented as $I(x+\delta x, y+\delta_y, z+\delta z, t+\delta t)$, where $\delta x, \delta y, \delta z$ is the motion of the vowel along the direction of 3 axes, respectively, so the following equation is satisfied,

$$I(x, y, z, t) = I(x + \delta x, y + \delta_y, z + \delta z, t + \delta t) \tag{4.1}$$

Based on 1st order Taylor series expansion, $I(x + \delta x, y + \delta_y, z + \delta z, t + \delta t)$ could be expanded as,

$$I(x + \delta x, y + \delta y, z + \delta z, t + \delta t) = I(x, y, z, t) + \frac{\partial I}{\partial x}\delta x + \frac{\partial I}{\partial y}\delta y + \frac{\partial I}{\partial z}\delta z + \frac{\partial I}{\partial t}\delta t \tag{4.2}$$

According to equation (4.1) and (4.2), the following equation could be obtained,

$$\frac{\partial I}{\partial x}v_x + \frac{\partial I}{\partial y}v_y + \frac{\partial I}{\partial z}v_z + \frac{\partial I}{\partial t} = 0 \tag{4.3}$$

Where v_x, v_y, v_z denotes the velocity of the voxel with respect to X, Y, Z axis, respectively, which is $v_x = \frac{\delta x}{\delta t}, v_y = \frac{\delta y}{\delta t}, v_z = \frac{\delta z}{\delta t}$. Let I_x, I_y, I_z, I_t denotes the 4 partial derivatives of I, which is, $I_x = \frac{\partial I}{\partial x}$, $I_y = \frac{\partial I}{\partial y}$, $I_z = \frac{\partial I}{\partial z}$, $I_t = \frac{\partial I}{\partial t}$ and equation (4.3) now becomes,

$$\nabla I \cdot \overrightarrow{\mathrm{v}} = -I_t \tag{4.4}$$

Where $\nabla I = [I_x, I_y, I_z] \in \mathbb{R}^3$ is the spatial intensity gradient and $\overrightarrow{\mathrm{v}} = [v_x, v_y, v_x]^T \in \mathbb{R}^3$ is the velocity of the image, in other words, the scene flow, of the pixel (x, y) at time t. Equation (4.4) is called the 3D Motion Constraint Equation.

4.1 Object Scene Flow (OSF) computation

A common approach for extracting 3D scene flow from 2D images is to use a stereo camera to extract the depth information of the pixels on the image, and thus it becomes able to compute the motion of pixels along Z axis. In this thesis, an approach, named object scene flow, is deployed in 3d scene flow computation for the stereo came [23]. This approach assumes that the 3D structure of the scene is approximated by a set of piecewise planar superpixels [34] and the moving object in the scene is rigid and finite. Let S denote the set of superpixels and $s \in S$ represents one of the superpixel in the set. Define the variable $v_s = [n_s, o_s] \in \mathbb{R}^2$, where $n_s \in \mathbb{R}^3$ denotes a 3d plane ($n_s \cdot X = 0$ for any point $X \in \mathbb{R}^3$ on the plane), and o_s denotes the objects that superpixel s is associated with. Define $\delta o_i \in SE^3$ as the motion of object o_i and O as the set of all the object, which is $O = \{o_1, o_2, ...\}$. So the scene flow could be computed by minimizing the cost function,

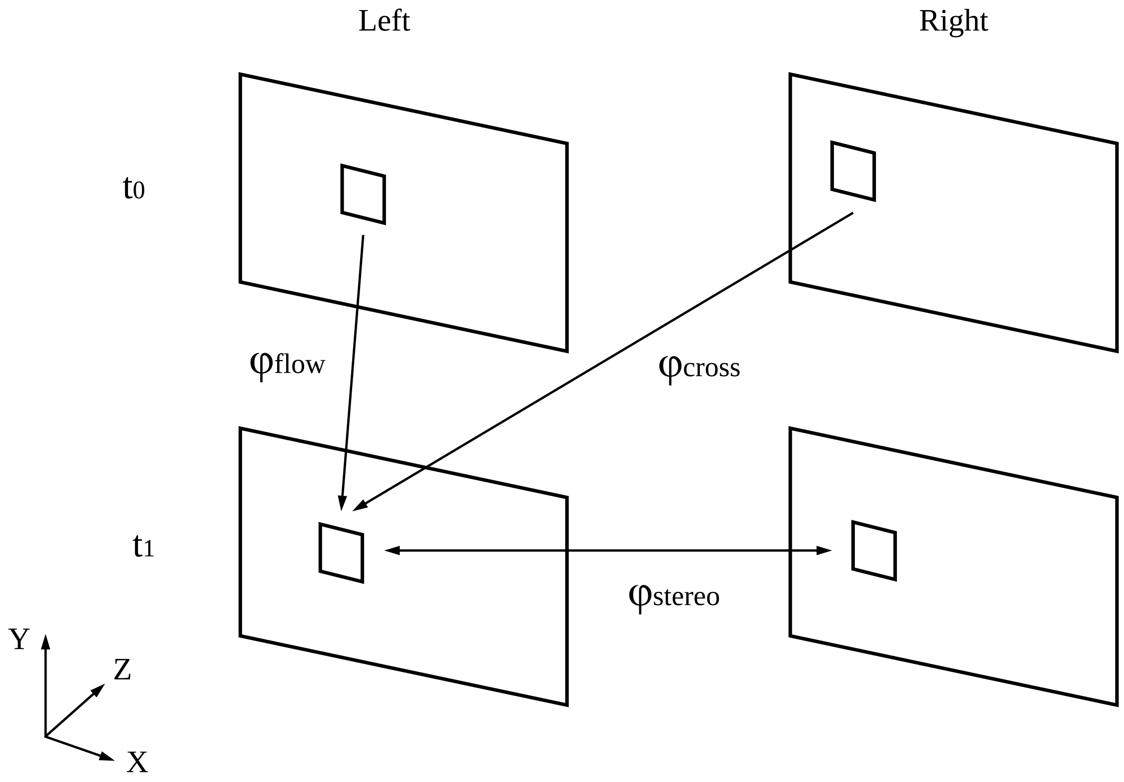
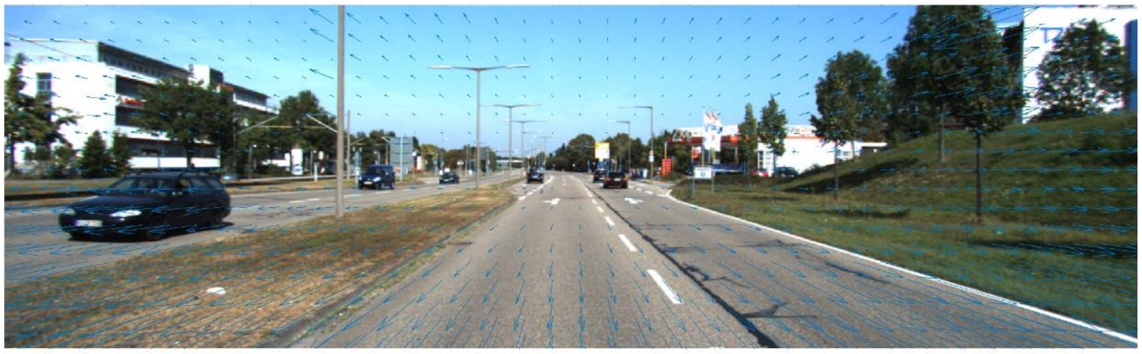


Figure 4.1: Data corresponding cost

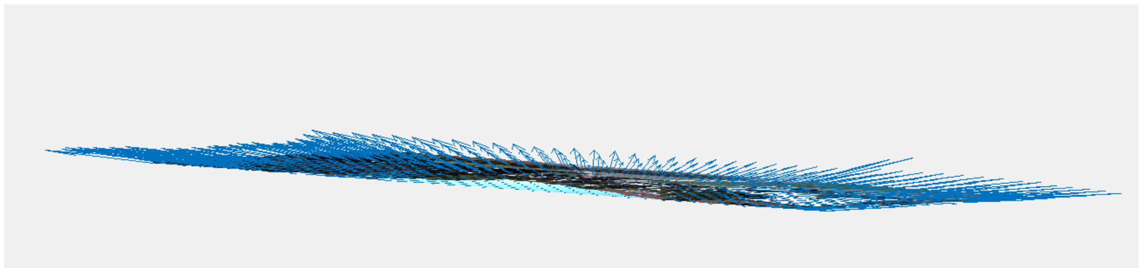
$$E(v_s, o_i) = \sum_{s \in S} \varphi(v_s, o_s) + \sum_{\{a|a \in S\, is adjacent to s\}} \psi(s, a) \tag{4.5}$$

where $\varphi(v_s, o_s) = \varphi_{flow}(v_s, o_s) + \varphi_(v_s, o_s)_{cross} + \varphi_{stereo}(v_s, o_s)$ denotes the cost of data corresponding, which consists of three parts, as figure 4.1 shows, and $\psi(s, a) = w_1\psi_{depth}(n_s, n_a)+w_2\psi_{orient}(n_s, n_a)+w_3\psi_{motion}(s, a)$ denotes the cost of smoothing, which consist of three part, the smoothness of depth, orientation and motion, respectively, between neighboring pixels, and w_1, w_2, w_3 denotes the weights to different parts.

Figure 4.2 shows a demo result of scene flow computation. In this demo the scene flow is presented as a set of arrows $A = \{A_i, i = 1, ...\} \in \mathbb{R}^3$ with the root of each arrow on each pixel $p_i(x, y, z)$of the image plane $P \in \mathbb{R}^3$, and each arrow



(a)



(b)

Figure 4.2: Demo result of scene flow computation, (a) projection of scene flow on the 2D image plane, (b) presentation of 3D scene flow arrow and the 2d image plane

presents the motion $\frac{dp_i(x,y,z)}{dt} \in \mathbb{R}^3$ of the corresponding pixels of the image.

4.1.1 3D reconstruction and visualization

To better present the 3D spatial information of the 3D scene flow, the 3D reconstruction are conducted, which projects the pixels on the 2D image back to the 3D space in the world coordinate, based on the depth information computed in the process of the scene flow computation. With stereo camera, suppose $I_1(x_1, y_1, t)$ and $I_2(x_2, y_2, t)$ are corresponding pixels respectively on two camera plane P_1 and P_2, where (x_1, y_1) and $(x_2, y_2) \in \mathbb{R}^2$ denotes the position of the pixel respectively in two camera's coordinates, and $I(x, y, z, t)$ where $(x, y, z) \in \mathbb{R}^3$ denotes the position of the voxel in the world coordinate, denotes the 3D voxel reconstructed from the pixel $I_1(x_1, y_1, t)$ and $I_2(x_2, y_2, t)$. Define a $\mathbb{R}^2 \times \mathbb{R}^2 \rightarrow \mathbb{R}^3$ projection function,

$$\begin{aligned}
&I(x, y, z, t) = p(I_1(x_1, y_1, t), I_2(x_2, y_2, t)) \\
&I_1(x_1, y_1, t) \in P_1 \\
&I_2(x_2, y_2, y) \in P_2
\end{aligned} \tag{4.6}$$

According to figure (4.3), the depth D of voxel $I(x, y, z, t)$ in the world coordinate could thus be computed as follows,

$$D = b\frac{f}{d},$$

where b is the stereo baseline and f denotes the focal length of camera lens, which are calibration parameters of the stereo camera, so in this thesis they are supposed to be known. The disparity $d = |x1 - x2|$ represents distance between the pixel positions in two camera coordinates. Now the scene could be reconstructed as a set V, which contains all the voxels projected from all the pixels of the image the camera captured, which is,

$$\begin{aligned}
&p(I_1(x1, y1, t), I_2(x2, y2, t)) = I(x1, y1, D, t) \\
&V = \{I \mid I = p(I_1, I_2), I_1 \in P_1, I_2 \in P_2\}
\end{aligned} \tag{4.7}$$

The result of 3D reconstruction of the scene and the scene flow visualization can be seen in figure (4.4). The image shows clearly the spatial relationship

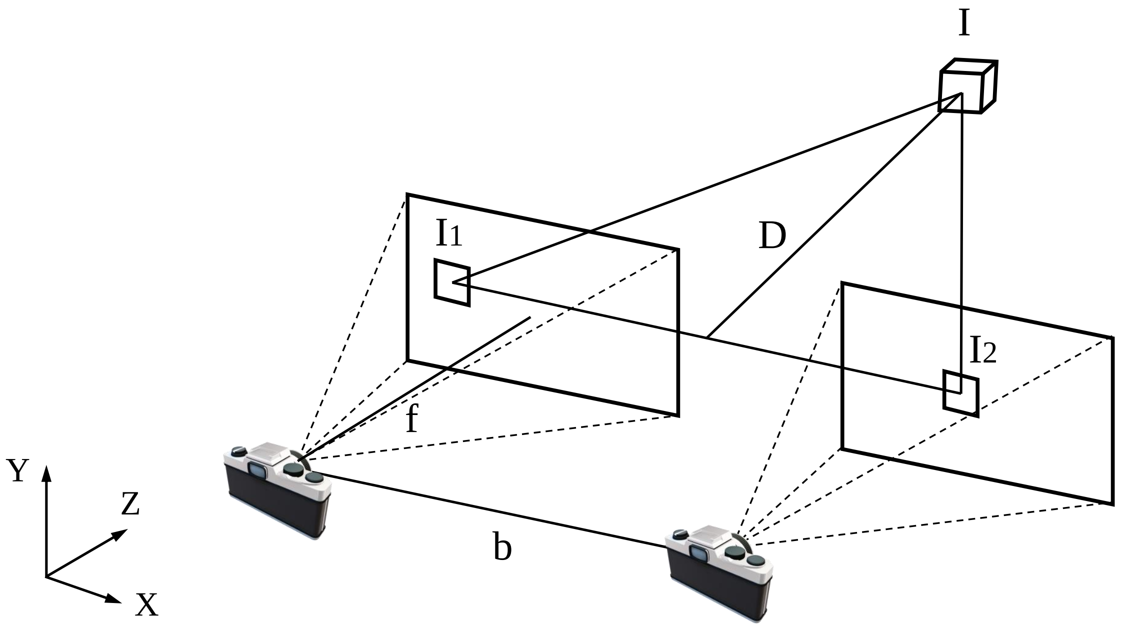
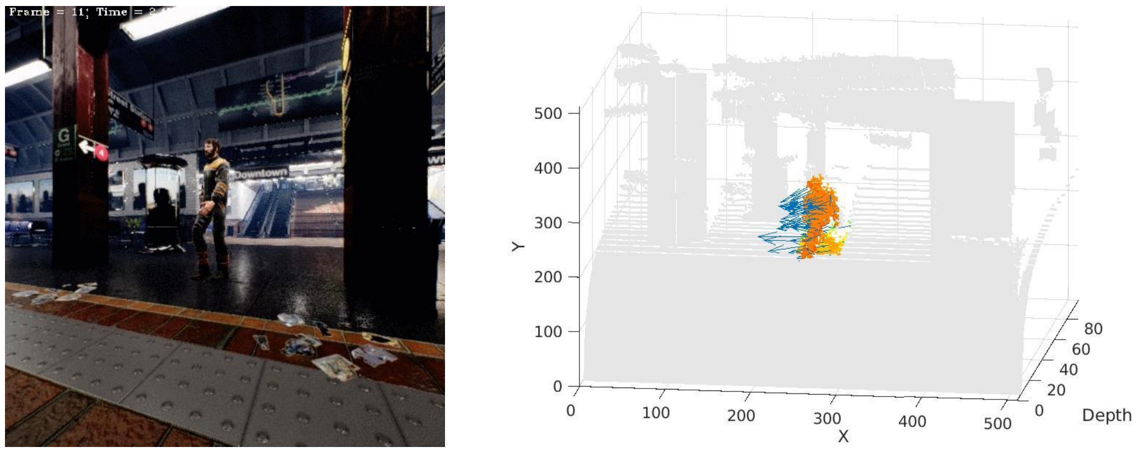


Figure 4.3: Sketch map of depth computation by stereo camera



(a) (b)

Figure 4.4: 3D reconstruction result. (a) the original image frame captured by the camera. (b) The reconstructed scene based on the 3D position of the pixel

among the person and the background, proving the algorithm to be efficient.

4.2 Ego-Motion Subtraction

Object scene flow is used to compute the dense flow of the whole scene. However in practical application only the motion of target, in the other word, sparse flows are needed. So it becomes necessary to compute the flow caused by ego motion of the camera and subtract it to generate the sparse flow.

Suppose the camera coordinate is fixed, define $I(x, y, z) = [x, y, z]^T$ as a voxel with the position $(x, y, z) \in \mathbb{R}^3$ denotes the position in the camera coordinate, λ denotes the focal length of the camera, and $I(p, q)$ denotes the pixel projected by the voxel on the image with the position $(p, q) \in \mathbb{R}^2$ in the camera coordinate, which is

$$I(p, q) = [p, q]^T = [\lambda \frac{x}{z}, \lambda \frac{y}{z}]^T \tag{4.8}$$

The derivative of function (4.8) is as following,

$$\nabla I(p, q) = [\lambda \frac{z \frac{\partial I(x,y,z)}{x} - x \frac{\partial I(x,y,z)}{z}}{z^2}, \lambda \frac{z \frac{\partial I(x,y,z)}{y} - y \frac{\partial I(x,y,z)}{z}}{z^2}]^T \tag{4.9}$$

The motion of the voxel with respect to the camera coordinates are caused by two factors, the translation and the rotation of the camera, which is,

$$\nabla I(x, y, z) = v_{trans} + v_{rot} \tag{4.10}$$

Define $w = [w_x, w_y, w_z]$ as the angular velocity of the camera around x, y, z axis respectively, $v = [v_x, v_y, v_z]$ as the linear velocity of the camera along x, y, z axis respectively, so now function (4.10) becomes,

$$\begin{aligned}\nabla I(x,y,z) &= v - w \times I(x,y,z) \\ &= [v_x + yw_z - zw_y, v_y + zw_x - xw_z, v_z + xw_y - yw_x]^T\end{aligned} \tag{4.11}$$

Based on function (4.9) and (4.11), the camera's ego motion could be represented as the motion of the voxel in the camera coordinate, which is,

$$\nabla I(x,y,z) = \begin{bmatrix} \frac{\lambda}{z} & 0 & -\frac{p}{z} & \frac{pq}{\lambda} & -\frac{p^2+\lambda^2}{\lambda} & q \\ 0 & \frac{\lambda}{z} & -\frac{p}{z} & -\frac{q^2+\lambda^2}{\lambda} & \frac{pq}{\lambda} & -p \\ 0 & 0 & 1 & -y & x & 0 \end{bmatrix} \begin{bmatrix} v & w \end{bmatrix}^T \tag{4.12}$$

To prove the efficiency of the ego motion subtraction algorithm, a demo is demonstrated using a moving camera to capture the images of static environment. The original scene flow computed is shown in figure (4.5.a) and scene flow after ego motion subtraction is shown in figure (4.5.b). As can be seen in the figures, the ego motion algorithm is able to compute the flow generated by camera motion and after subtraction the flow are much more closed to the ground truth (no flow).

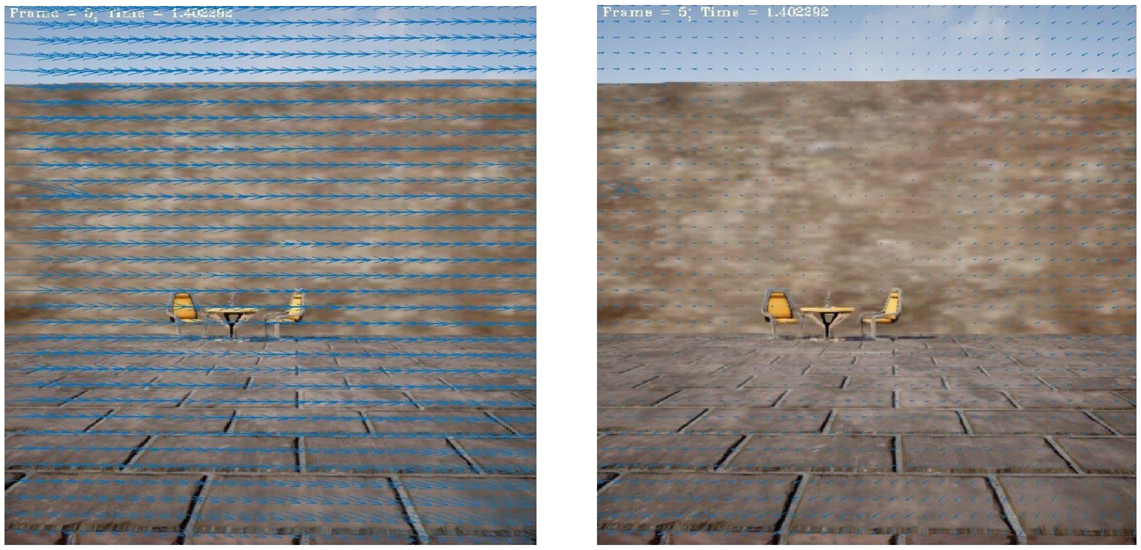


Figure 4.5: Demo result of scene flow ego motion subtraction. (a) Original scene flow computed from the images extracted by a moving camera (b) scene flow after camera ego motion subtraction.

CHAPTER 5

GAUSSIAN PROCESS ON MOTION PREDICTION

Recently, Bayesian non-parametric approaches began to increasingly show its effectiveness in learning dynamics from data since they are flexible and data driven. [22, 32, 20]. In this thesis, Bayesian non-parametric GP is deployed to model the velocity field for each target, which takes the target's position as the input and the corresponding velocity of that target as the output.

In particular, let $\mathbf{X}_i(k) = [\mathbf{x}_i(1) \quad \dots \quad \mathbf{x}_i(k)]$ and $\mathbf{Z}_{ij}(k) = \left[\mathbf{z}_{ij}(1) \quad \dots \quad \mathbf{z}_{ij}(k)\right]$ represent the measured positions and velocities of ith target. The jth sensor's local GP model would predict the velocity of certain target at any query position $\boldsymbol{\xi} \in \mathcal{W}$, and the predicted value $\mathbf{z}_{ij}(\boldsymbol{\xi})$ would obey the following Gaussian distribution [28],

$$\begin{aligned}
&\mathbf{z}_{ij}(\boldsymbol{\xi}) \sim \mathcal{N}\left(\boldsymbol{\mu}_{ij}(\boldsymbol{\xi}), \Sigma_{ij}(\boldsymbol{\xi})\right) \\
&\boldsymbol{\mu}_{ij}(\boldsymbol{\xi}) = \mathbf{K}(\boldsymbol{\xi}, \mathbf{X})\left(\mathbf{K}(\mathbf{X}, \mathbf{X}) + \epsilon_0^2 \mathbf{I}\right)^{-1} \mathbf{Z} \\
&\Sigma_{ij}(\boldsymbol{\xi}) = \mathbf{K}(\boldsymbol{\xi}, \boldsymbol{\xi}) - \mathbf{K}(\boldsymbol{\xi}, \mathbf{X})\left(\mathbf{K}(\mathbf{X}, \mathbf{X}) + \epsilon_0^2 \mathbf{I}\right)^{-1} \mathbf{K}(\mathbf{X}, \boldsymbol{\xi})
\end{aligned} \tag{5.1}$$

where $\boldsymbol{\mu}_{ij}(\boldsymbol{\xi})$ and $\Sigma_{ij}(\boldsymbol{\xi})$ respectively represent the mean and covariance matrix of the Gaussian distribution.

To encode the similarity between input data points, a kernel was added to the computation during the GP process. In this thesis, the following Radial Basis functions matrix $\mathbf{K}(\cdot, \cdot)$ is deployed as the spatio-temporal kernel to account for the time-varying nature of the motion model,

$$\mathbf{K}\left(\mathbf{x}_i(t_i), \mathbf{x}_j(t_j)\right) = \sigma_s^2 e^{-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|_2^2}{2l_x^2}} e^{-\frac{(t_i - t_j)^2}{2l_t^2}} \tag{5.2}$$

where l_x and l_t respectively denotes the spatial and temporal length scale, and σ_s represents the hyper-parameter for the signal variance.

5.1 Local GP Learning and Prediction of Target Trajectory

For sensor j, the hyper-parameters of GP could be learned from the measurement by maximizing the logarithm of the marginal likelihood function of the training data. [28]. Then the resultant GP model could be used to predict the target position within the time interval $\left[k, k_f\right]$ during the planning process. According to Bayesian's rule, the probability density function (pdf) of the predicted target positions could be represented as,

$$P_j\left(\mathbf{X}_i(k+1:k_f) \mid \mathbf{X}_i(k)\right) = \prod_{t=k}^{k_f-1} P\left(\mathbf{x}_i(t+1) \mid \mathbf{x}_i(t)\right) \tag{5.3a}$$

$$= \prod_{t=k}^{k_f-1} \int_{\mathbb{R}^2} P\left(\mathbf{x}_i(t+1) \mid \mathbf{v}_i(t), \mathbf{x}_i(t)\right) P_j\left(\mathbf{v}_i(t) \mid \mathbf{x}_i(t)\right) d\mathbf{v}_i(t), \tag{5.3b}$$

where $P_j\left(\mathbf{v}_i(t) \mid \mathbf{x}_i(t)\right)$ corresponds to jth sensor's local GP model and $P\left(\mathbf{x}_i(t+1) \mid \mathbf{v}_i(t), \mathbf{x}_i(t)\right)$ is defined by the target motion model (3.1).The factorizatoin in (5.3a) is due to the Markov property of the target motion model. In general, there is no analytical form for $P_j\left(\mathbf{X}_i(k+1:k_f) \mid \mathbf{X}_i(k)\right)$ when $k_f - k \geq 2$.

To make the target motion prediction tractable, in this thesis a *nominal path* is defined, which is obtained by assuming that the target is moving in a mean velocity given from the GP model by approximating the posterior probability $P_j\left(\mathbf{X}_i(k+1:k_f) \mid \mathbf{X}_i(k)\right)$ along the nominal path. Above process essentially resembles the Laplace approximation which is widely used in statistical inference [4]. In particular, define the sequence of targets' nominal positions as $\hat{\mathbf{X}}_{ij}(k+1:k_f) = [\hat{\mathbf{x}}_{ij}(k+1) \quad \ldots \quad \hat{\mathbf{x}}_{ij}(k_f)]$ where $\hat{\mathbf{x}}_{ij}(t+1) = \boldsymbol{\mu}_{ij}(\hat{\mathbf{x}}_{ij}(t))\Delta T + \hat{\mathbf{x}}_{ij}(t)$, $t = k, \ldots, k_f - 1$ with the initial condition $\hat{\mathbf{x}}_{ij}(k) = \mathbf{x}_i(k)$. The velocity term $\boldsymbol{\mu}_{ij}(\mathbf{x}_i(t))$ is the mean vector computed from (5.1). Then the pdf of the predicted trajectory can be approximated as follows,

$$
\begin{aligned}
&P_j\left(\mathbf{X}_i(k+1:k_f) \mid \mathbf{X}_i(k)\right) && \text{(5.4a)}\\
&\qquad \approx \prod_{t=k}^{k_f-1} \int_{\mathbb{R}^2} P\left(\mathbf{x}_i(t+1) \mid \mathbf{v}_i(t), \hat{\mathbf{x}}_{ij}(t)\right) P_j\left(\mathbf{v}_i(t) \mid \hat{\mathbf{x}}_{ij}(t)\right) d\mathbf{v}_i(t)\\
&\qquad = \prod_{t=k}^{k_f-1} P_j\left(\mathbf{v}_i(t) = \frac{\mathbf{x}_i(t+1) - \hat{\mathbf{x}}_{ij}(t)}{\Delta T} \mid \hat{\mathbf{x}}_{ij}(t)\right) && \text{(5.4b)}
\end{aligned}
$$

where the equality (5.4b) holds since the motion model (3.1) is deterministic. Equations (5.4a) indicates that the pdf of the predicted target trajectory could be approximated by computing the pdf of the predicted velocities along the nominal path, which is a product of Gaussian distributions. By simple algebraic manipulation, it could be shown that (5.4b) is actually a Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}_{ij,loc}, \boldsymbol{\Sigma}_{ij,loc})$, where the mean vector $\boldsymbol{\mu}_{ij,loc}$ and the covariance matrix $\boldsymbol{\Sigma}_{ij,loc}$ is

$$
\boldsymbol{\mu}_{ij,loc} = \left[\hat{\mathbf{x}}_{ij}(k+1) \quad \ldots \quad \hat{\mathbf{x}}_{ij}(k_f)\right]^T, \tag{5.5a}
$$

$$
\boldsymbol{\Sigma}_{ij,loc} = diag\left[\boldsymbol{\Sigma}_{ij}^T(\hat{\mathbf{x}}_{ij}(k+1)) \quad \ldots \quad \boldsymbol{\Sigma}_{ij}^T(\hat{\mathbf{x}}_{ij}(k_f))\right], \tag{5.5b}
$$

where *diag* means the block diagonal matrix. It can be easily seen the mean is the vector of nominal positions.

5.2 Decentralized Target Trajectory Fusion and Prediction

Since the sensing range is limited (and usually smaller than the workspace), for each sensor, only a subset of the targets can be measured at each time step. So to coordinate the whole sensing maps, the local prediction of each sensor need to be fused so that a global consensus on the prediction of targets' trajectories can be obtained.

In this subsection, a rumor-robust decentralized GP fusion approach is proposed. Consider the fusion process of ith target's prediction from sensors j and l, where the pdfs of local prediction are $P_j\left(\mathbf{X}_i(k+1:k_f) \mid \mathbf{X}_i(k)\right)$ and $P_l\left(\mathbf{X}_i(k+1:k_f) \mid \mathbf{X}_i(k)\right)$, which could be computed using (5.4a). Then the fused pdf could be obtained from the following formula,

$$\begin{aligned} P\left(\mathbf{X}_i(k+1:k_f)|\mathbf{X}_i(k)\right) \propto\ & P^{\beta_j w^*}\left(\mathbf{X}_{ij}(k+1:k_f)|\mathbf{X}_{ij}(k)\right) \\ & P^{\beta_l(1-w^*)}\left(\mathbf{X}_{il}(k+1:k_f)|\mathbf{X}_{il}(k)\right), \end{aligned} \tag{5.6}$$

where β_j and β_l are the weighting factors that indicate each sensor's contribution to the combined prediction, and w^* is the optimal weight based on Chernoff information [25]. According to the strategy in [6], β_j and β_l are chosen as the difference in the differential entropy between the prior and posterior at $\mathbf{X}_i(k+1 : k_f)$ to ensure that the more information an agent contains on the ith target's prediction, the more it could contribute to the combined prediction. Using the fact that for a Gaussian distribution $P(\mathbf{x}) \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, its exponential is also a Gaussian distribution with a scaled covariance matrix [6], i.e. $P^{\alpha}(\mathbf{x}) \sim \mathcal{N}(\boldsymbol{\mu}, \alpha^{-1}\boldsymbol{\Sigma})$, the predictive mean and covariance matrix of $P\left(\mathbf{X}_i(k+1:k_f) \mid \mathbf{X}_i(k)\right)$ are

$$\begin{aligned}
\boldsymbol{\mu}_i^* &= \boldsymbol{\Sigma}_i^* \left(\beta_j w^* \boldsymbol{\Sigma}_{ij,loc}^{-1} \boldsymbol{\mu}_{ij,loc} + \beta_l (1 - w^*) \boldsymbol{\Sigma}_{il,loc}^{-1} \boldsymbol{\mu}_{il,loc}\right) \\
\boldsymbol{\Sigma}_i^* &= \left(\beta_j w^* \boldsymbol{\Sigma}_{ij,loc}^{-1} + \beta_l (1 - w^*) \boldsymbol{\Sigma}_{il,loc}^{-1}\right)^{-1}
\end{aligned} \tag{5.7}$$

where $\left(\boldsymbol{\mu}_{ij,loc}, \boldsymbol{\Sigma}_{ij,loc}\right)$ and $\left(\boldsymbol{\mu}_{il,loc}, \boldsymbol{\Sigma}_{il,loc}\right)$ are respectively the mean and covariance pairs of sensor j and l 's local prediction.

The fusion rule proposed in (5.6) is similar to the generalized product of GP experts (gPoE) method as proposed in [6]. However, since these methods assume that for each agent, the training data are disjoint, therefore directly multiplying local pdfs to generate the glocal pdf is a reasonable choice in [6].

However, while in the practical application, common information can usually exist in the training data of different sensors in a sensor network, which means that a target could be measured simultaneously by multiple sensors. So how to avoid double-counting becomes an essential requirement for the consistent data fusion in the sensors networks [5]. To avoid rumor propagation, in this thesis, weighted exponential product rule are utilized for data fusion. This thesis deploys Chernoff information to compute the optimal fusion weight, which has been proved effective to reduce rumor propagation in sensor networks since the common information would only be counted exactly once via weighted exponential production.[1]. For two arbitrary pdfs $P_a(\mathbf{x})$ and $P_b(\mathbf{x})$, the optimal

Chernoff weight is obtained by minimizing their Chernoff information, i.e.,

$$w^* = \arg\max_{w\in[0,1]} -\log \int [P_a(\mathbf{x})]^w \, [P_b(\mathbf{x})]^{1-w} \, d\mathbf{x}. \tag{5.8}$$

The combined pdf is $P(\mathbf{x}) = [P_a(\mathbf{x})]^{w^*} \, [P_b(\mathbf{x})]^{1-w^*}$.

The main difficulty of using Chernoff weight for information fusion is that for general distributions, there is usually no analytical expression for their Chernoff information, therefore computing the optimal weight may cause significant computational cost [1]. However, the Chernoff information for two multivariate Gaussian distributions, $P_a(\mathbf{x}) \sim \mathcal{N}(\boldsymbol{\mu}_a, \boldsymbol{\Sigma}_a)$ and $P_b(\mathbf{x}) \sim \mathcal{N}(\boldsymbol{\mu}_b, \boldsymbol{\Sigma}_b)$, could be expressed in a closed form [25], and the optimal Chernoff weight can be computed as follows,

$$\begin{aligned} w^* = \arg\min_{w\in[0,1]} & \frac{1}{2} \log \frac{|w\boldsymbol{\Sigma}_a + (1-w)\boldsymbol{\Sigma}_b|}{|\boldsymbol{\Sigma}_a|^w |\boldsymbol{\Sigma}_b|^{1-w}} \\ + & \frac{w(1-w)}{2} (\boldsymbol{\mu}_a - \boldsymbol{\mu}_b)^T (w\boldsymbol{\Sigma}_a + (1-w)\boldsymbol{\Sigma}_b) (\boldsymbol{\mu}_a - \boldsymbol{\mu}_b) \end{aligned} \tag{5.9}$$

The optimal Chernoff weight, w^*, in (5.7) therefore can be computed efficiently for the following two Gaussian distributions,

$$P^{\beta_j}\left(\mathbf{X}_{ij}(k+1:k_f) \mid \mathbf{X}_{ij}(k)\right) \sim \mathcal{N}(\boldsymbol{\mu}_{ij,loc}, \beta_j^{-1}\boldsymbol{\Sigma}_{ij,loc}), \tag{5.10}$$

$$P^{\beta_l}\left(\mathbf{X}_{il}(k+1:k_f) \mid \mathbf{X}_{il}(k)\right) \sim \mathcal{N}(\boldsymbol{\mu}_{il,loc}, \beta_l^{-1}\boldsymbol{\Sigma}_{il,loc}), \tag{5.11}$$

using nonlinear optimization algorithms [26]. Using (5.7) and (5.9) for each pair of sensors along the tree-structured communication network, the rumor-robust decentralized GP fusion can therefore be conducted efficiently. The fused prediction at the root sensor could then be propagated back to all sensors such that all the sensors would have the same fused pdf, which is refered to as $P_{fuse}\left(\mathbf{X}_i(k+1:k_f) \mid \mathbf{X}_i(k)\right)$ for the ith target.

In a summary, he process of the fusion could be described by the following pseudocode,

Algorithm 1 Decentralized GP model fusion

```
1: Let $\mathbf{S} = [\mathbf{S}_1 \dots \mathbf{S}_m]$ be the set of sensors
2: Let $\mathbf{X} = [\mathbf{X}_1 \dots \mathbf{X}_n]$ be the set of targets
3: for target $\mathbf{X}_i$ in $\mathbf{X}$ do
4:     Let $\mathbf{X}_i(k) = [\mathbf{x}_i(1) \dots \mathbf{x}_i(k)]$ be the set of position of target $x_i$ in time frames 1 to $n$
5:     for sensor $\mathbf{S}_j$ in $\mathbf{S}$ do
6:         Let $\mathbf{Z}_{ij}(k) = \left[\mathbf{z}_{ij}(1) \dots \mathbf{z}_{ij}(k)\right]$ be the set of the velocity of the target $\mathbf{X_i}$ in time frames $1-k$ measured by the sensor $\mathbf{S}_j$
7:         Compute the GP model $P_j\left(\mathbf{X}_i(k+1:k_f) \mid \mathbf{X}_i(k)\right)$ of the target $\mathbf{X_i}$ in sensor $\mathbf{S}_j$
8:         if $j > 1$ then
9:             Fuse the two GP models $P_{j-1}\left(\mathbf{X}_i(k+1:k_f) \mid \mathbf{X}_i(k)\right)$ and $P_j\left(\mathbf{X}_i(k+1:k_f) \mid \mathbf{X}_i(k)\right)$ stored in $\mathbf{S}_j$ into the new GP model $P_j^*\left(\mathbf{X}_i(k+1:k_f) \mid \mathbf{X}_i(k)\right)$
10:            Update the GP models $P_j\left(\mathbf{X}_i(k+1:k_f) \mid \mathbf{X}_i(k)\right)$ in sensor $\mathbf{S}_j$ to the fused GP model $P_j^*\left(\mathbf{X}_i(k+1:k_f) \mid \mathbf{X}_i(k)\right)$
11:        end if
12:        Pass the GP model $P_j\left(\mathbf{X}_i(k+1:k_f) \mid \mathbf{X}_i(k)\right)$ in sensor $\mathbf{S_j}$ to sensor $\mathbf{S_{j+1}}$
13:    end for
14:    for sensor $\mathbf{S}_j$ in $\mathbf{S}$ do
15:        Pass the GP model $P_n\left(\mathbf{X}_i(k+1:k_f) \mid \mathbf{X}_i(k)\right)$ in sensor $\mathbf{S_m}$ to sensor $\mathbf{S_j}$
16:        Update the GP model $P_j\left(\mathbf{X}_i(k+1:k_f) \mid \mathbf{X}_i(k)\right)$ in sensor $\mathbf{S}_j$ to GP model $P_n\left(\mathbf{X}_i(k+1:k_f) \mid \mathbf{X}_i(k)\right)$
17:    end for
18: end for
```

CHAPTER 6

PATH PLANNING VIA INFORMATION ENTROPY

In this section, a decentralized IPP algorithm using the sequential planning strategy will be presented. [17, 2, 7]. According to [17, 2], compared to the centralized planning, the sequential planning is guaranteed to generate a near-optimal solutions, provided that the objective function is submodular and monotonic and satisfies MI. For sequential planning, after the planning order is given, each sensor would at first receive the planning information from its predecessors (Section 6.1), and then compute its own optimal path (Section 6.2). Later each sensor would passing the new planning information to the next sensor in the sequence. The planning order can be defined in advance or determined online for each communication round. In this thesis, it's assumed that the planning order corresponds to the sensors' indices. The decentralized IPP algorithm is characterized by the efficiency both in communication (Section 6.1) and computation (Section 6.2). Since mutual information is a submodular function, such sequential planning approach would generate a near-optimal, coordinated solution [16].

6.1 Fusing Predecessors' Plans

For the jth sensor, given the planned paths of the first $j-1$ sensors, $\mathbf{S}_{j-1}(k:k_f) = [\mathbf{s}_1(k:k_f) \quad \dots \quad \mathbf{s}_{j-1}(k:k_f)]$, the local planning problem then becomes

$$
\begin{aligned}
\mathbf{u}_j^*\left(k:k_f\right) = \arg\max_{\mathbf{u}_j\left(k:k_f\right)} J_i\left(\mathbf{u}_j^*\left(k:k_f\right) \mid \mathbf{S}_{j-1}(k:k_f)\right) \\
s.t.\ \mathbf{s}_j(t+1) = \mathbf{g}\left(\mathbf{s}_j(t), \mathbf{u}_j(t)\right), \\
\mathbf{s}_j(t) \in \mathcal{S},\ \mathbf{u}_j(t) \in \mathcal{U},\ t = k, \ldots, k_f
\end{aligned}
\tag{6.1}
$$

The objective function is defined as the mutual information between target prediction and jth sensor's planned path, conditioned on the first $j-1$ sensors' plans, i.e.,

$$
J_j\left(\mathbf{u}_j^*\left(k:k_f\right) \mid \mathbf{S}_{j-1}(k:k_f)\right) = I\left(\mathbf{X}\left(k+1:k_f\right); \mathbf{z}_j\left(k+1:k_f\right) | \mathbf{Z}_{j-1}(k+1:k_f)\right) \tag{6.2}
$$

where $\mathbf{Z}_{j-1}(k+1:k_f)$ represent the predicted measurements of the first $j-1$ sensors' planned paths.

The main difficulty in evaluating (6.2) is how to systematically update the target position prediction by incorporating predecessors' planned path. To solve this problem, a Bayesian data fusion strategy is deployed for fusing predecessors' plan.

In order to fuse predecessor sensors' predicted measurements, the globally fused target prediction, $P_{fuse}\left(\mathbf{X}_i(k+1:k_f) \mid \mathbf{X}_i(k)\right)$, is treated as the prior distribution of targets' prediction. Similar to Section 5.1, the Bayesian fusion approach is used to compute the posterior distribution conditioned on $\mathbf{Z}_{j-1}(k+1:k_f)$. Define $\hat{\mathbf{X}}_i\left(k+1:k_f\right) = [\hat{\mathbf{x}}_i(k+1) \quad \ldots \quad \hat{\mathbf{x}}_i(k_f)]$ as the nominal path obtained from $P_{fuse}\left(\mathbf{X}_i(k+1:k_f) \mid \mathbf{X}_i(k)\right)$, the pdf after fusing $\mathbf{Z}_{j-1}(k+1:k_f)$ is

$$
\begin{aligned}
&P_{j,pre}\left(\mathbf{X}_i\left(k+1:k_f\right) \mid \mathbf{Z}_{j-1}(k+1:k_f)\right) \\
&\propto P_{fuse}\left(\mathbf{X}_i\left(k+1:k_f\right)\right)\prod_{l=1}^{j-1} P\left(\mathbf{z}_l(k+1:k_f) \mid \mathbf{X}_i\left(k+1:k_f\right)\right) && \text{(6.3a)} \\
&\approx \prod_{t=k}^{k_f-1}\left(P\left(\mathbf{v}_i(t) = \frac{\mathbf{x}_i(t+1)-\hat{\mathbf{x}}_i(t)}{\Delta T}\right)\prod_{l=1}^{j-1} P\left(\mathbf{z}_l(t) \mid \hat{\mathbf{x}}_i(t)\right)\right) && \text{(6.3b)}
\end{aligned}
$$

The factorization in (6.3a) is obtained by the conditional independence of measurements from different sensors given the target positions. Equation (6.3b) is obtained similar to that in (5.4a), where the pdf is approximated along the nominal path. The predicted measurement from lth sensor, $\mathbf{z}_l(t)$, is assumed to be nonempty if the nominal position $\hat{\mathbf{x}}_i(t)$ lies in the sensor's FOV at t.

The prior $P\left(\mathbf{v}_i(t) = \frac{\mathbf{x}_i(t+1)-\hat{\mathbf{x}}_i(t)}{\Delta T}\right)$ can be directly obtained by marginalizing $P_{fuse}\left(\mathbf{X}_i(k+1:k_f) \mid \mathbf{X}_i(k)\right)$ over all time steps except t, and can be easily shown to be a Gaussian distribution, denoted as $\mathcal{N}(\boldsymbol{\mu}_{i,fuse}(t), \boldsymbol{\Sigma}_{i,fuse}(t))$. Since the measurement model is linear Gaussian, an analytical expression of $P_{j,pre}\left(\mathbf{X}_i\left(k+1:k_f\right) \mid \mathbf{Z}_{j-1}(k+1:k_f)\right)$ can be obtained. In particular, let $\mathbb{I}\{\hat{\mathbf{x}}_i(t) \in \mathcal{F}(\mathbf{s}_l(t))\}$ represent the indicator function and it equals 1 if and only if the the nomial position $\hat{\mathbf{x}}_i(t)$ lies in the sensor's planned FOV at t. Then it can be derived, using the conjugacy property of Gaussian prior and likelihood functions [4], that given the prior covariance matrix $\boldsymbol{\Sigma}_{i,fuse}(t)$ and let $n(t) = \sum_{l=1}^{j-1} \mathbb{I}\{\hat{\mathbf{x}}_i(t) \in \mathcal{F}(\mathbf{s}_l(t))\}$ represent the number of sensors in the first $j-1$ sensors that can measure the ith target at time t, then the posterior covariance is

$$
\boldsymbol{\Sigma}_{ij,pre}(t) = \left(\boldsymbol{\Sigma}^{-1}_{i,fuse}(t) + n(t)\boldsymbol{\Sigma}^{-1}_{\varepsilon}\right)^{-1}, \; t = k+1,\ldots,k_f. \tag{6.4}
$$

Therefore, the fused pdf can be compactly represented as

$$
\begin{aligned}
P_{j,pre}\left(\mathbf{X}_i\left(k+1:k_f\right) \mid \mathbf{Z}_{j-1}(k+1:k_f)\right) &\sim \mathcal{N}\left(\boldsymbol{\mu}_{ij,pre}(k+1:k_f), \boldsymbol{\Sigma}_{ij,pre}(k+1:k_f)\right) \\
&= \prod_{t=k+1}^{k_f} \mathcal{N}\left(\boldsymbol{\mu}_{ij,pre}(t), \boldsymbol{\Sigma}_{ij,pre}(t)\right)
\end{aligned}
\tag{6.5}
$$

where the mean and the covariance matrix $\boldsymbol{\Sigma}_{ij,pre}(k+1:k_f)$ is

$$
\begin{aligned}
\boldsymbol{\mu}_{ij,pre}\left(k+1:k_f\right) &= \left[\boldsymbol{\mu}^T_{ij,pre}(k+1) \quad \dots \quad \boldsymbol{\mu}^T_{ij,pre}(k_f)\right]^T \\
\boldsymbol{\Sigma}_{ij,pre}\left(k+1:k_f\right) &= diag\left[\boldsymbol{\Sigma}_{ij,pre}(k+1) \quad \dots \quad \boldsymbol{\Sigma}_{ij,pre}(k_f)\right].
\end{aligned}
\tag{6.6}
$$

The key observation is that, to compute $\boldsymbol{\Sigma}_{ij,pre}\left(k+1:k_f\right)$, the only information needed from predecessor sensors is the times that each sensor are expected to detect the target during the planning interval, i.e., $n(t)$. Since all sensors are sharing the same nominal path of the target, $\hat{\mathbf{X}}_i\left(k:k_f\right)$, with the decentralized GP fusion, what each sensor needs to do is only to receive the counting number from its predecessor, then update its own counting number by adding the received number to its expected measurement times. After updating its GP, the sensor would then send the updated counting number to the next sensor. The communication between each pair of sensors is therefore constant and independent of the number of predecessor sensors. In contrast, in state-of-the-art sequential planning approaches [17, 2, 7], the transmitted information to each sensor is the planned path from all predecessor sensors, which has the communication burden of $O(N)$. So the decentralized sequential planning approach in this thesis could significantly reduce the communication cost compared to the state-of-the-art.

6.2 Local Objective Function

The fused pdf $P_{j,pre}\left(\mathbf{X}_i\left(k:k_f\right) \mid \mathbf{Z}_{j-1}(k:k_f)\right)$ can now be used as the prior pdf for jth sensor's path planning. Given the jth sensor's future control inputs $\mathbf{u}_j\left(k:k_f\right)$ and the consequent future measurements $\mathbf{z}_j\left(k:k_f\right)$, the posterior pdf can be obtained in a way similar to (6.3), which is,

$$\begin{aligned}
&P_{j,plan}\left(\mathbf{X}_i\left(k+1:k_f\right) \mid \mathbf{z}_j\left(k+1:k_f\right), \mathbf{Z}_{j-1}(k+1:k_f)\right)\\
&\propto P_{j,pre}\left(\mathbf{X}_i\left(k+1:k_f\right) | \mathbf{Z}_{j-1}(k+1:k_f)\right) \qquad (6.7)\\
&\quad P\left(\mathbf{z}_j\left(k+1:k_f\right) | \mathbf{X}_i\left(k+1:k_f\right)\right)\\
&\approx \prod_{t=k+1}^{k_f} P\left(\hat{\mathbf{v}}_i(t) = \frac{\mathbf{x}_i(t+1) - \hat{\mathbf{x}}_i(t)}{\Delta T} \mid \mathbf{Z}_{j-1}(k+1:k_f)\right)\\
&\quad P\left(\mathbf{z}_j(t) \mid \hat{\mathbf{x}}_i(t)\right)\\
&\sim \mathcal{N}\left(\boldsymbol{\mu}_{ij,plan}(k+1:k_f), \boldsymbol{\Sigma}_{ij,plan}(k+1:k_f)\right),
\end{aligned}$$

where the covariance matrix is

$$\boldsymbol{\Sigma}_{ij,plan}\left(k+1:k_f\right) = diag\left[\boldsymbol{\Sigma}_{ij,plan}(k+1) \quad \ldots \quad \boldsymbol{\Sigma}_{ij,plan}(k_f)\right]. \qquad (6.8)$$

Again, using the conjugacy of Gaussian distribution, the covariance matrix of the posterior pdf could be computed in a closed-form, i.e., for $t = k+1, \ldots, k_f$,

$$\boldsymbol{\Sigma}_{ij,plan}(t) = \left(\boldsymbol{\Sigma}^{-1}_{ij,pre}(t) + \mathbb{I}\{\hat{\mathbf{x}}_i(t) \in \mathcal{F}\left(\mathbf{s}_j(t)\right)\}\boldsymbol{\Sigma}_{\varepsilon}^{-1}\right)^{-1}. \qquad (6.9)$$

Because the mutual information in (6.2) is a non-additive function of the future measurements, $\mathbf{z}_j\left(k+1:k_f\right)$, maximizing $J_j\left(\mathbf{u}_j\left(k:k_f\right) \mid \mathbf{S}_{j-1}(k:k_f)\right)$ is in general a combinatorial optimization problem since there is possibility that the predicted target might be outside of sensor FOV and therefore results in no measurement at some time steps. So a computationally efficient approach which is able to significantly reduce the computational complexity is developed.

Now derive the closed-form of the objective function. Let $H(\cdot)$ represent the entropy of random variables, then

$$\begin{aligned}
&J_j\left(\mathbf{u}_j\left(k+1:k_f\right) \mid \mathbf{S}_{j-1}(k+1:k_f)\right)\\
&=\sum_{i=1}^{M} I\left(\mathbf{X}_i\left(k+1:k_f\right);\mathbf{z}_j\left(k+1:k_f\right) \mid \mathbf{Z}_{j-1}(k+1:k_f)\right) &\text{(6.10a)}\\
&=\sum_{i=1}^{M}\sum_{t=k+1}^{k_f} I\left(\mathbf{X}_i\left(t\right);\mathbf{z}_j\left(t\right) \mid \mathbf{Z}_{j-1}(k+1:k_f)\right) &\text{(6.10b)}
\end{aligned}$$

where (6.10a) is due to the independence of GP models for different targets and (6.10b) is due to the block diagonal shape of $\mathbf{\Sigma}_{ij,plan}$. For a Gaussian distribution $P_a(\mathbf{x}) \sim \mathcal{N}(\boldsymbol{\mu}_a, \mathbf{\Sigma}_a)$, its entropy has a closed form, $H(\mathbf{x}) = \frac{1}{2}\log\det(2\pi e\mathbf{\Sigma}_a)$. Given the analytic expression of MI between Gaussian distributions [4], it can be derived that

$$J_j\left(\mathbf{u}_j^*\left(k:k_f\right) \mid \mathbf{S}_{j-1}(k:k_f)\right)=\sum_{i=1}^{M}\sum_{t=k}^{k_f}\frac{1}{2}\log\det\frac{\mathbf{\Sigma}_{ij,pre}(t)}{\mathbf{\Sigma}_{ij,plan}(t)}. \tag{6.11}$$

The indicator function in (6.9) makes the IPP problem a mixed integer nonlinear programming problem, which is notoriously difficult to solve. To overcome this problem, an approximate objective function is used, where the indicator function is replaced by constant 1 and a weighting factor is added to the

MI at each step. In particular, define the weighting factor

$$\psi(t) = \max\left(0, 1 - \frac{\left(\|[s_{j,x}(t), s_{j,y}(t)]^T - \hat{\mathbf{x}}_i(t)\|_2 - \frac{r_j}{2}\right)^2}{(\frac{r_j}{2})^2}\right). \tag{6.12}$$

which equals 1 when the sensor is close to the predicted target location, and equals 0 when they are far away from each other. Then the objective function is defined as

$$J_j\left(\mathbf{u}_j\left(k : k_f\right) \mid \mathbf{S}_{j-1}(k : k_f)\right) = \sum_{i=1}^{M}\sum_{t=k}^{k_f} \frac{\psi(t)}{2} \log\det \frac{\mathbf{\Sigma}_{ij,pre}(t)}{\tilde{\mathbf{\Sigma}}_{ij,plan}(t)} \tag{6.13}$$

where $\tilde{\mathbf{\Sigma}}_{ij,plan}(t) = \left(\mathbf{\Sigma}^{-1}_{ij,pre}(t) + \mathbf{\Sigma}^{-1}_{\varepsilon}\right)^{-1}$. By using above closed-form objective function, the decentralized IPP problem (6.1) is now efficiently solved with non-linear optimization algorithms.

CHAPTER 7

SIMULATION AND RESULT

The implementation of the decentralized GP fusion and information-driven path planning (IPP) algorithm based on the sparse optical flow of the target, as is described in the previous chapters, are tested in a variety of workspaces. Two kind of testing are conducted. One is to evaluate the accuracy of of the Gaussian Process prediction by stationary sensors in the workspace measuring the motion of the targets and thus making predictions. Another one is to evaluate the performance of the IPP, conducted by setting multiple sensors chasing multiple targets with different motion patterns in a larger workspace.

7.1 Evaluating Decentralized GP Learning and Fusion

To evaluate the accuracy and efficiency of the decentralized GP fusion algorithm, firstly set a $10m \times 10m$ workspace, 4 sensors are randomly set in the workspace. Set the range of the sensor as $r_j = 5m, j = 1 \dots N$. A target X_1 was randomly set in this workspace and moves along the curve of a differentiable function $f : y = \frac{139}{6600}x^3 - \frac{1393}{2200}x^2 + \frac{1729}{330}x$ in this workspace, shown as figure (7.1) and the time interval between each measurement of the sensors is $0.2s$. The result could be seen in figure (7.2) and (7.3), where figure (7.2) shows the state-of-art result of the decentralized GP fusion prediction, and figure (7.3) represents the analytical error of the prediction with respect to the ground truth. The result proves the accuracy of the decentralized GP fusion algorithm. However, to further evaluate the efficiency of decentralized GP fusion algorithm over the ordinary GP algorithm without decentralized fusion, another experiment needs

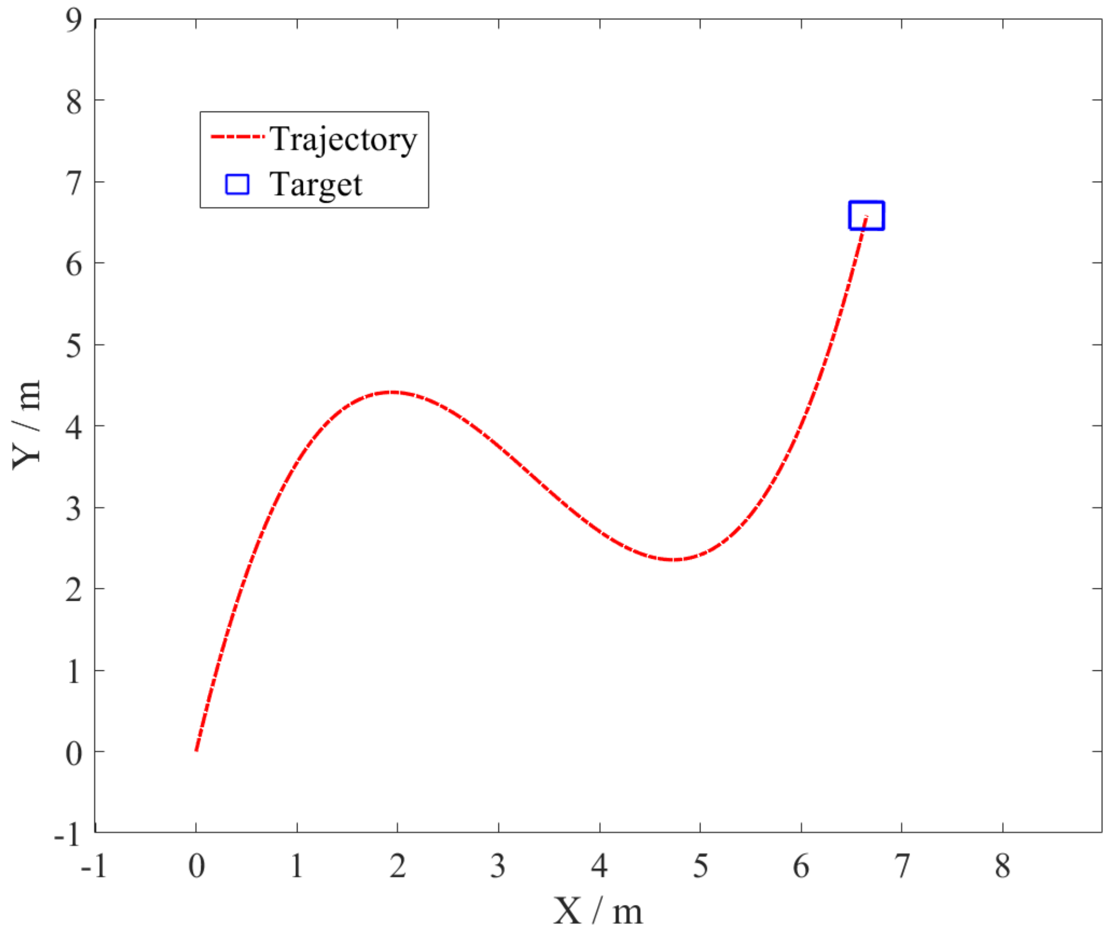


Figure 7.1: Trajectory of the target in the workspace

to be conducted.

Using the same workspace and sensors, to show the efficiency of the decentralized GP fusion algorithm, a set of 8 targets were randomly placed in the workspace with each target in a different motion pattern, the layout could be seen in figure (7.4). Figure (7.5) represents the trajectories of different targets. Three method, decentralized GP fusion, GP without fusion and centralized GP are tested in the simulation. In centralized GP, all the sensor are allowed to communicate and share the measurement between each other such that the GP learning and prediction are based the measurement of all the sensors, which

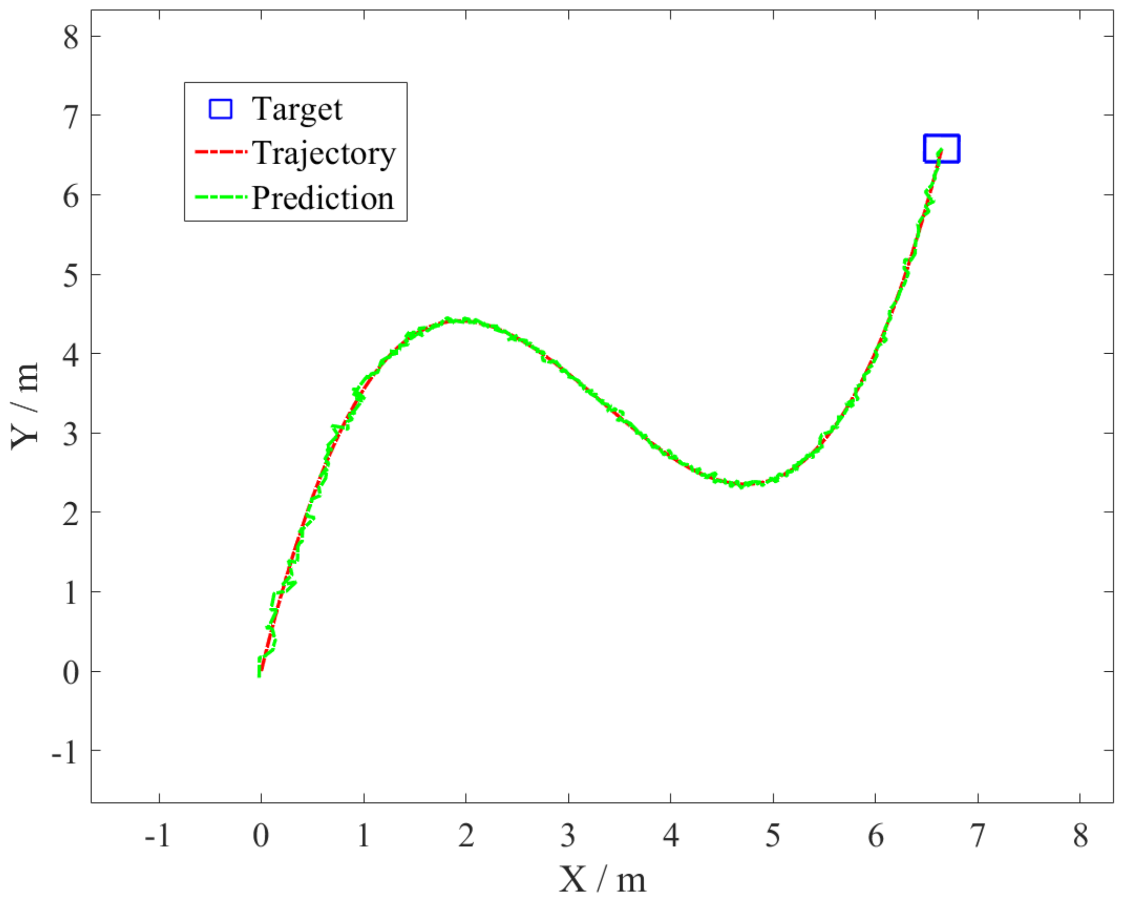


Figure 7.2: Result of decentralized GP fusion prediction of the target trajectory

stands for the optimal situation. GP without fusion, on the other hand, are basically set under the similar constrain to the decentralized GP fusion algorithms, that is, the communication between sensors are limited. For comparison, in GP without fusion, the sensors only uses their own measurement and do not communicate with other sensors at all.

The results could be seen in figure (7.6). As expected, the centralized GP represents the best performance among the three approaches. Since in the centralized GP, there is no limit of communication between sensors so all the sensors' measurement can be therefore used for learning the GP model. As a result, the error of centralized GP should be the lowest. Decentralized GP Fusion, shows

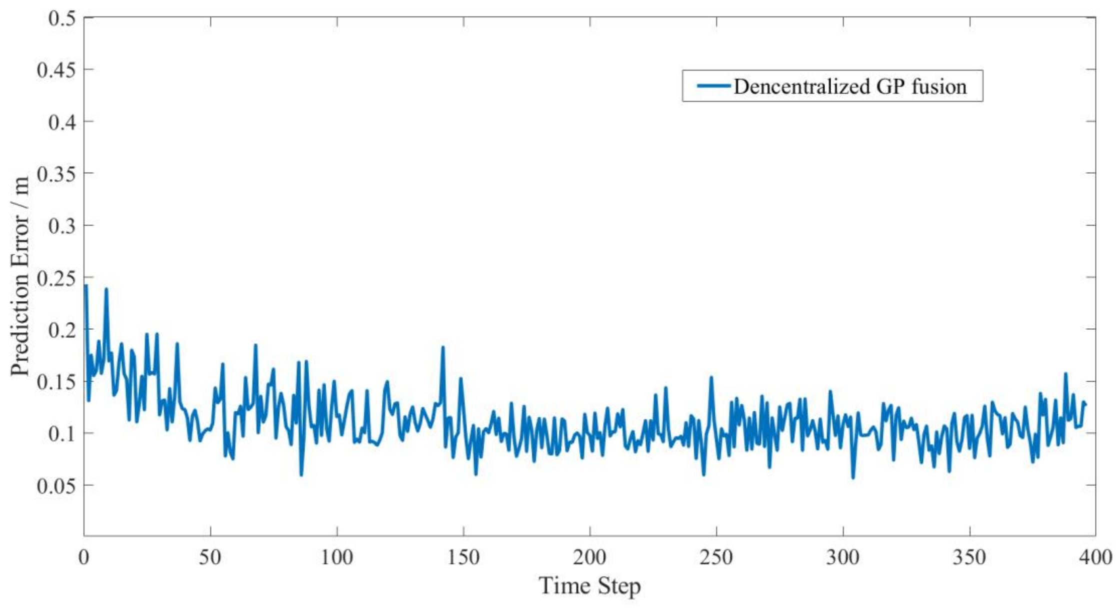


Figure 7.3: Decentralized GP fusion prediction error

a slightly larger error over the optimal situation while GP without fusion made a prediction with the largest error as each sensor is limited to only use its own measurement and there are no communications between sensors at all. This simulation results shows that decentralized GP fusion is efficient in the learning and prediction of the targets' motions with a relatively low prediction error over the ordinary GP without fusion and could achieve a similar accuracy with the centralized GP, while significantly reducing the computational complexity compared to centralized GP.

7.2 Evaluating the information-driven path planning algorithm (IPP)

In this simulation, to represent the efficiency of the path planning algorithm, a larger workspace with $30m \times 30m$ area was used but still, the sensors' measuring

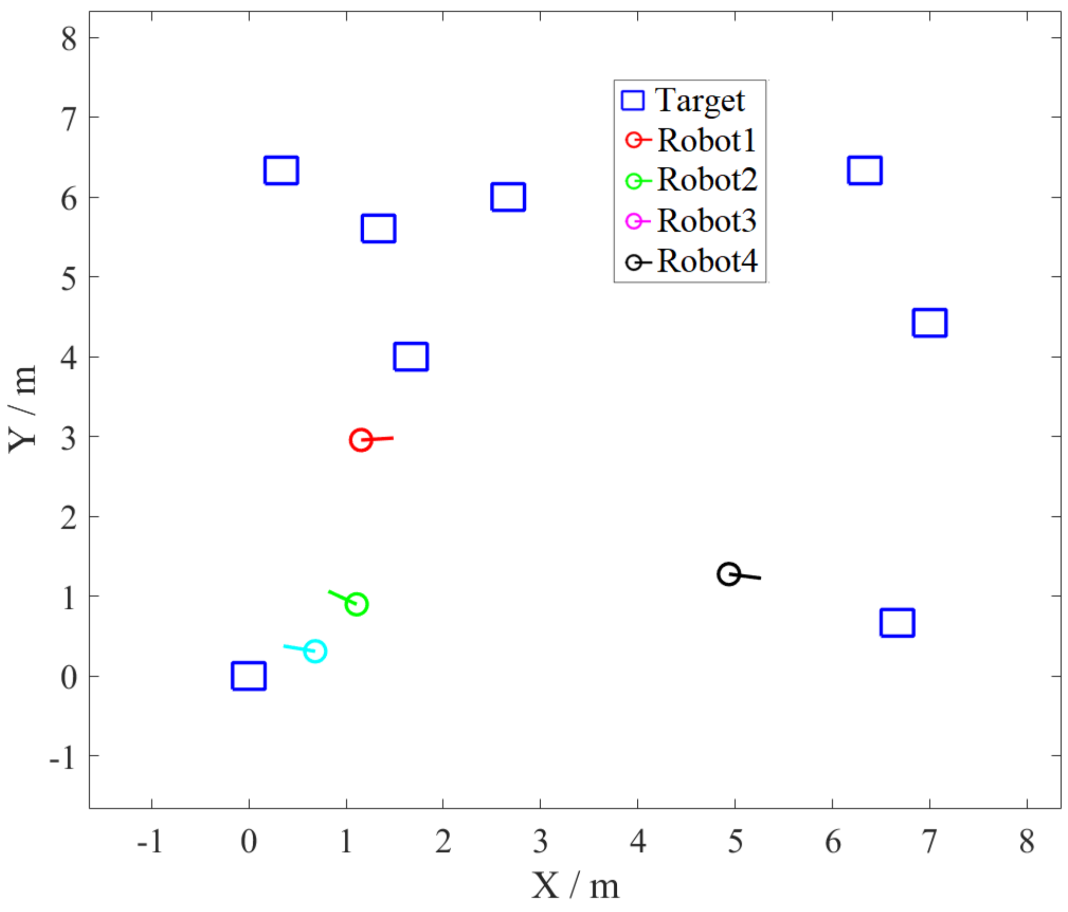


Figure 7.4: Initial layout of the simulation

range is limited to $r_j = 5m, j = 1 \dots N$. This setting ensures that the sensors are not able to detect all the targets at one time step. So the information communication and planning based on the analysis of past information are therefore significant. Multiple sensors, each installed on a mobile robot, are deployed to learn from the targets' motion patterns and based on that, chase the targets in order to get as much information from the targets as possible. The planning horizon is five steps ahead. The velocity of the mobile robot is bounded in the the range of $[0, 3](m/s)$ and the control input to each robot is also bounded in the range of,

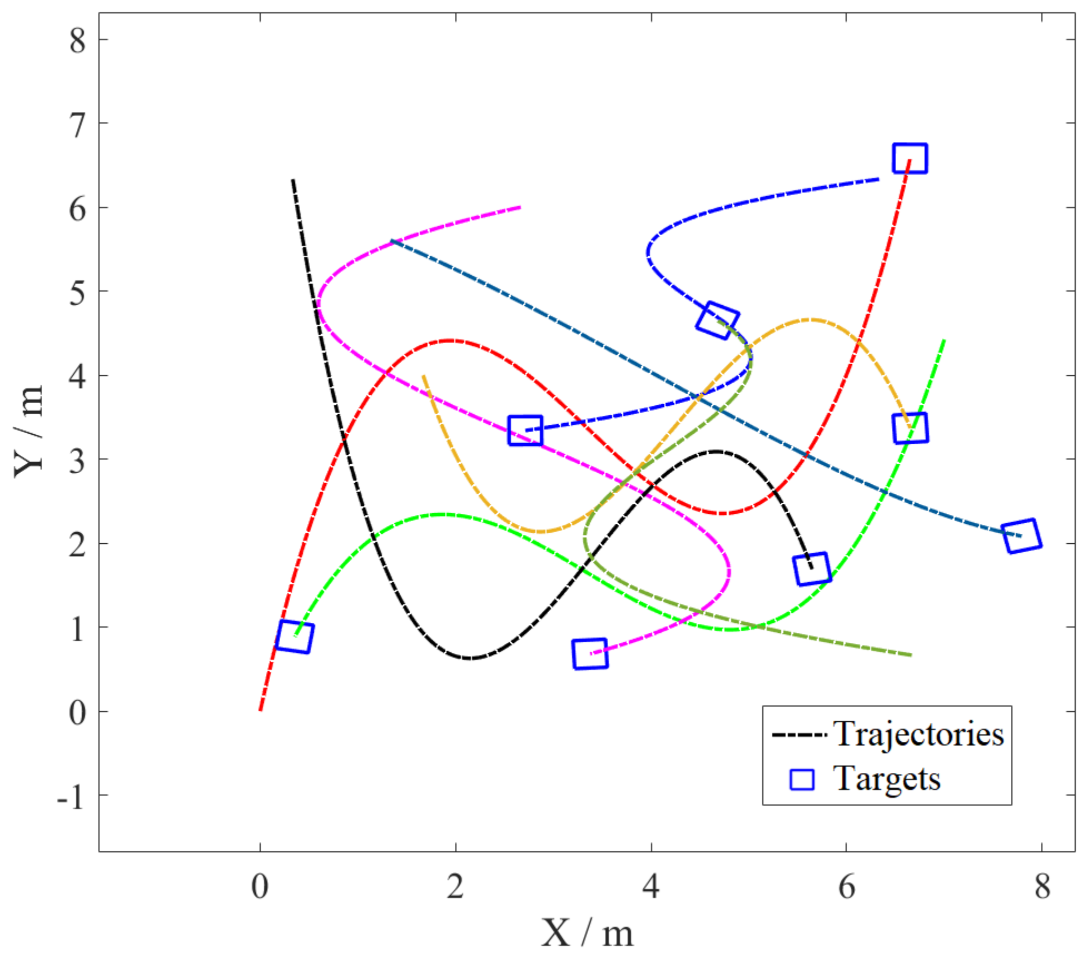


Figure 7.5: Trajectories of different targets

$$\begin{bmatrix} -\frac{\pi}{6} \\ -5(m/s) \end{bmatrix} \leq \mathbf{u} \leq \begin{bmatrix} \frac{\pi}{6} \\ 5(m/s) \end{bmatrix}.$$

Similar to the previous section, one target was placed in the workspace and moves along the curve of a differentiable function $f : x = \frac{97}{3360}y^3 - \frac{473}{560}y^2 + \frac{2629}{420}y + \frac{22}{35}$ in this workspace, as figure (7.7) shows. 2 sensors, which are randomly placed in the workspace, are deployed for measuring and chasing the target to show that IPP works in such situations. The result can be seen in figure (7.8) and (7.9), where figure (7.8) shows the visualized trajectories of the target and sensors to

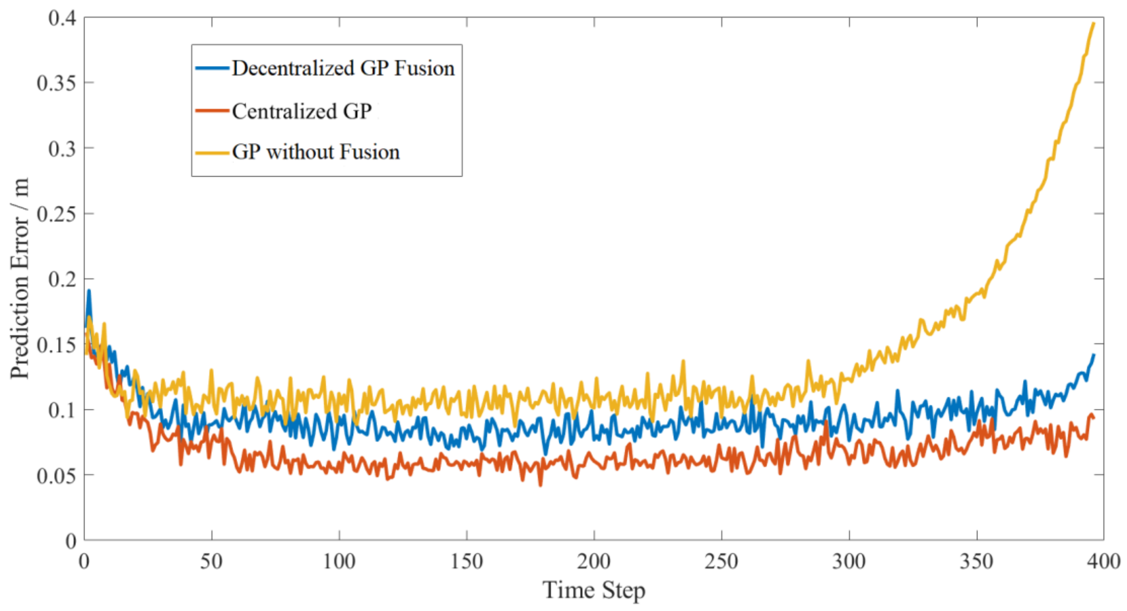


Figure 7.6: Average prediction error of targets using different GP fusion strategies.

show that the sensors are capable of chasing the target based on the GP models they learned and figure (7.9) represents the numerical error during the chasing process. It can be seen in the pictures that the IPP algorithm is efficient in chasing the target moving in the workspace.

7.3 Evaluating the decentralized GP learning, fusion, and planning algorithm (RESIN)

Now, the final simulation is going to be conducted to represent the performance of the whole algorithm proposed in this thesis, while compared with other algorithms. The settings are like the following. In a $30m \times 30m$ workspace, 8 targets are randomly placed and each with different motion patterns. 4 sensors, with each installed on a mobile robot, are also randomly placed in the workspace.

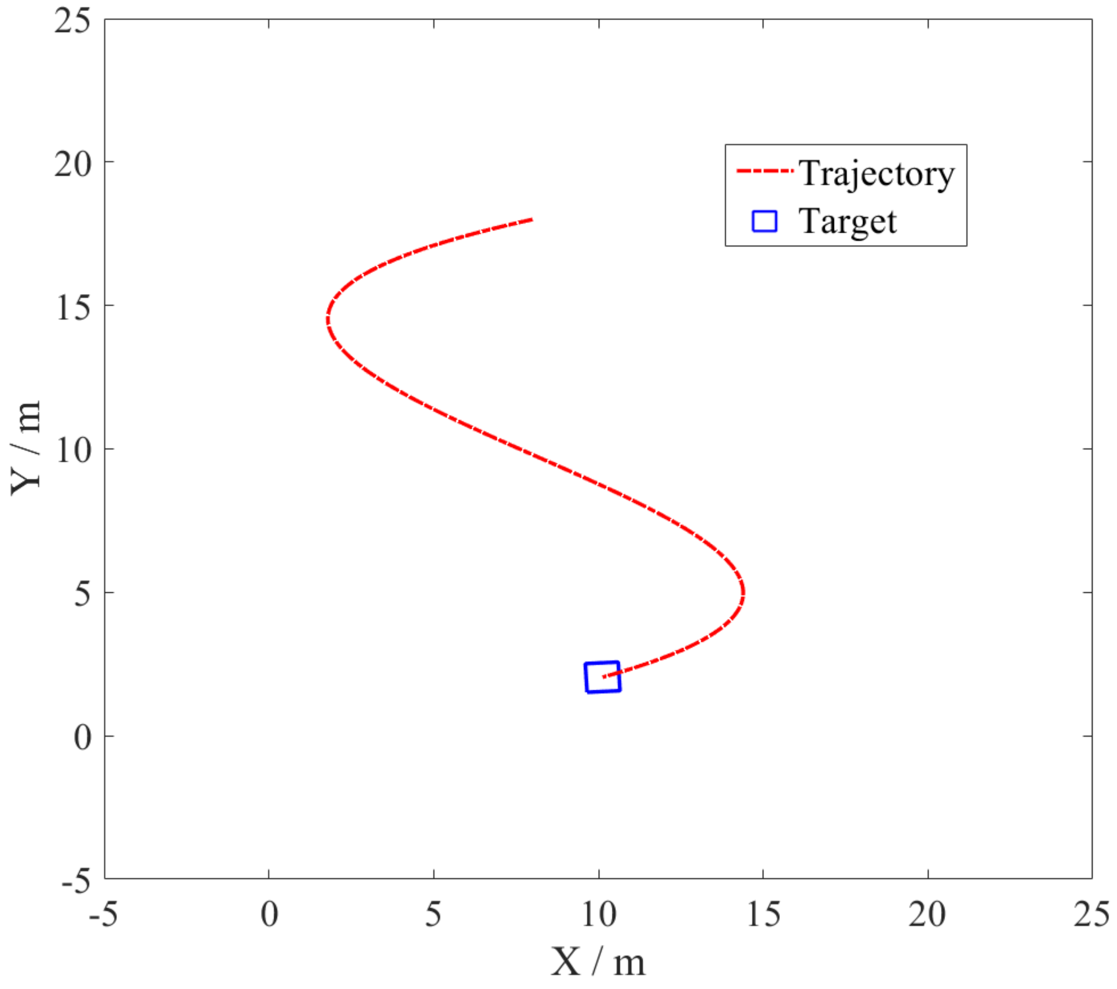


Figure 7.7: Trajectory of the target in the workspace

The initial layout could be seen in figure (7.10). The measuring range of each sensor is $r_j = 5m, j = 1 \ldots N$ and the velocity of the robots is in the the range of $[0, 3](m/s)$. The planning horizon is five steps ahead. The control input into the robot is,

$$\begin{bmatrix} -\frac{\pi}{6} \\ -5(m/s) \end{bmatrix} \leq \mathbf{u} \leq \begin{bmatrix} \frac{\pi}{6} \\ 5(m/s) \end{bmatrix}.$$

Four algorithms are used, including RESIN and three other algorithms, centralized GP planner, nearest target following planner, and the random planner.

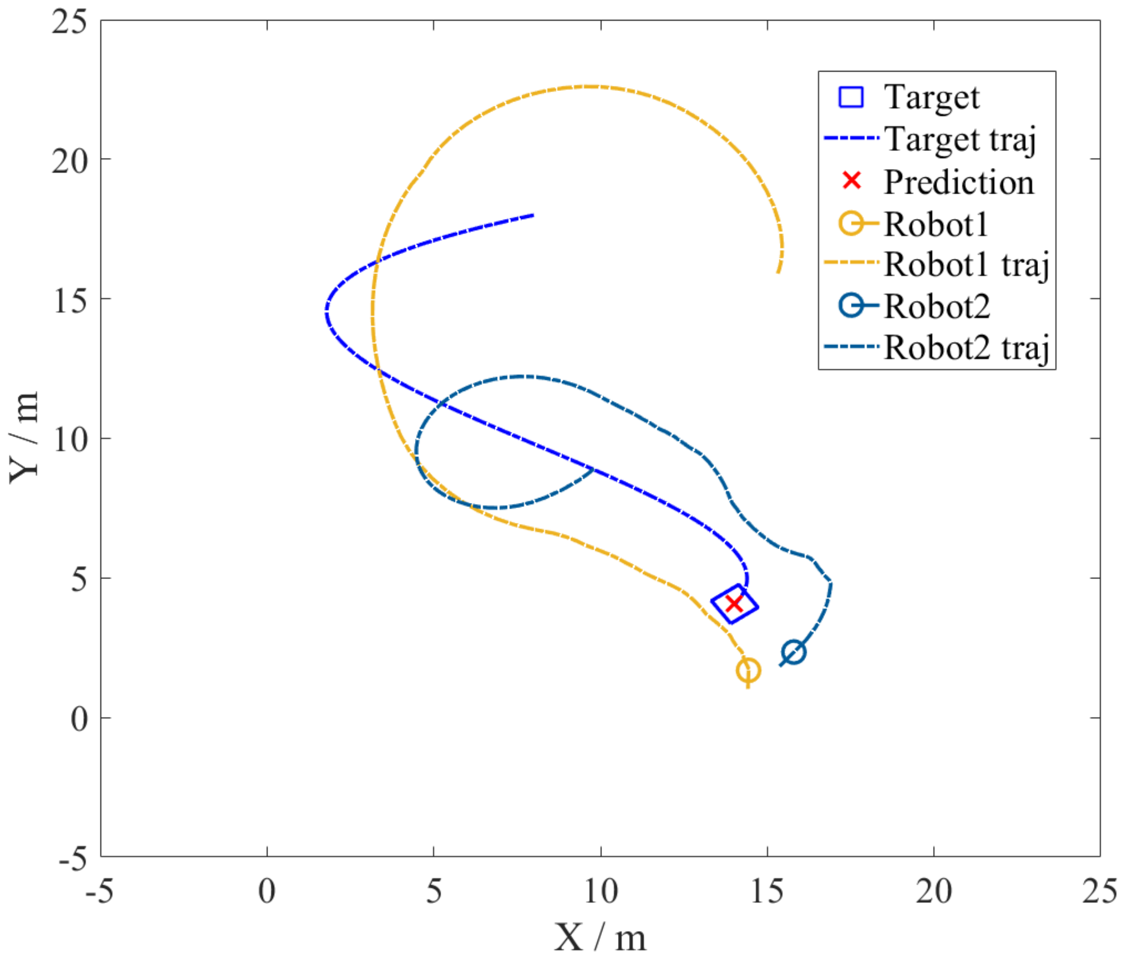


Figure 7.8: Result of robots chasing the target and prediction on the target position

In the centralized GP planner, for comparison. In the centralized GP planner, sensors' communication are not limited and all the sensors could share their measurement with each other, so the planning are conducted in a centralized way, which is based on all the sensors' data and plans for all the sensors at the same time. In the nearest target following planner, each sensor would pursue the nearest target from itself based on its local prediction of the targets' positions. The random planner would not make use of the sensor measurement at all and would just generate random control inputs for each sensor.

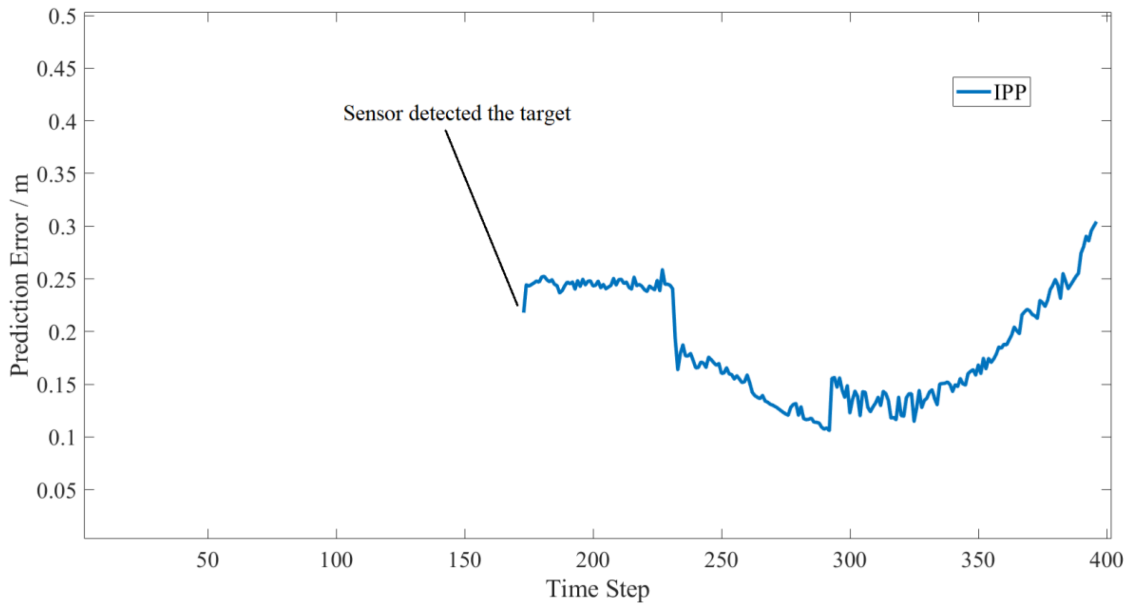


Figure 7.9: Result of prediction error of target position

The result could be seen in figure (7.11) and (7.12). Figure (7.11) visually represents the performance of the 4 algorithms. As expected, the centralized GP planner achieved the highest performance since all sensor's measurements and synthesized and are used for the planning of all the sensors simultaneously. RESIN shows a very similar and closed performance with the centralized GP planner, for which the sensors are able to keep track of targets and make accurate prediction of the targets' position. In contrast, in the nearest target following planner simulation, several targets are lost track of and the prediction on these targets becomes inaccurate. Random planner performs the worst, since the sensors do not take their measurement into account at all, the target easily lose the track of their targets very soon. The numerical results shown in figure (7.12) further support the observation, in which the average prediction error on each target by each sensor under different planning strategies are quantitatively compared. From the picture it is to see that RESIN outperforms the nearest target following planner and random planner and the performance is very closed

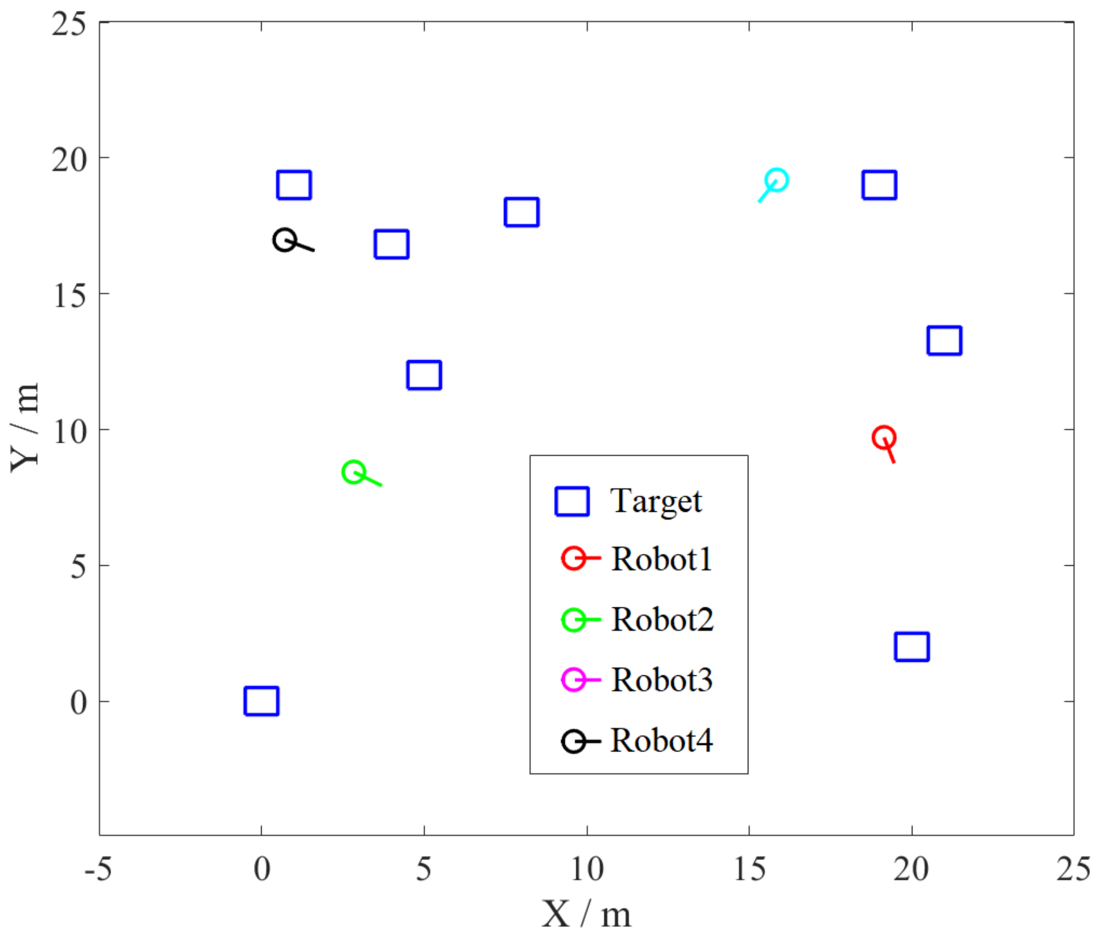


Figure 7.10: Initial layout of the simulation

to the performance of the optimal situation, the centralized GP planner.

In order to further explore into the applicability of RESIN. the workspace and initial setting would be kept unchanged but the number of sensors deployed for measuring and tacking the targets is changing. The number of targets is 8 and the number of sensors (robots) changed from 2 to 8. The result could be seen in figure (7.13) and table (7.1). The result shows that with the increasing of the number of the sensors, the difference of the prediction error between RESIN and centralized GP has been enlarging. With 8 sensors, the performance of RESIN is even worse than nearest target following planner. Though with 2

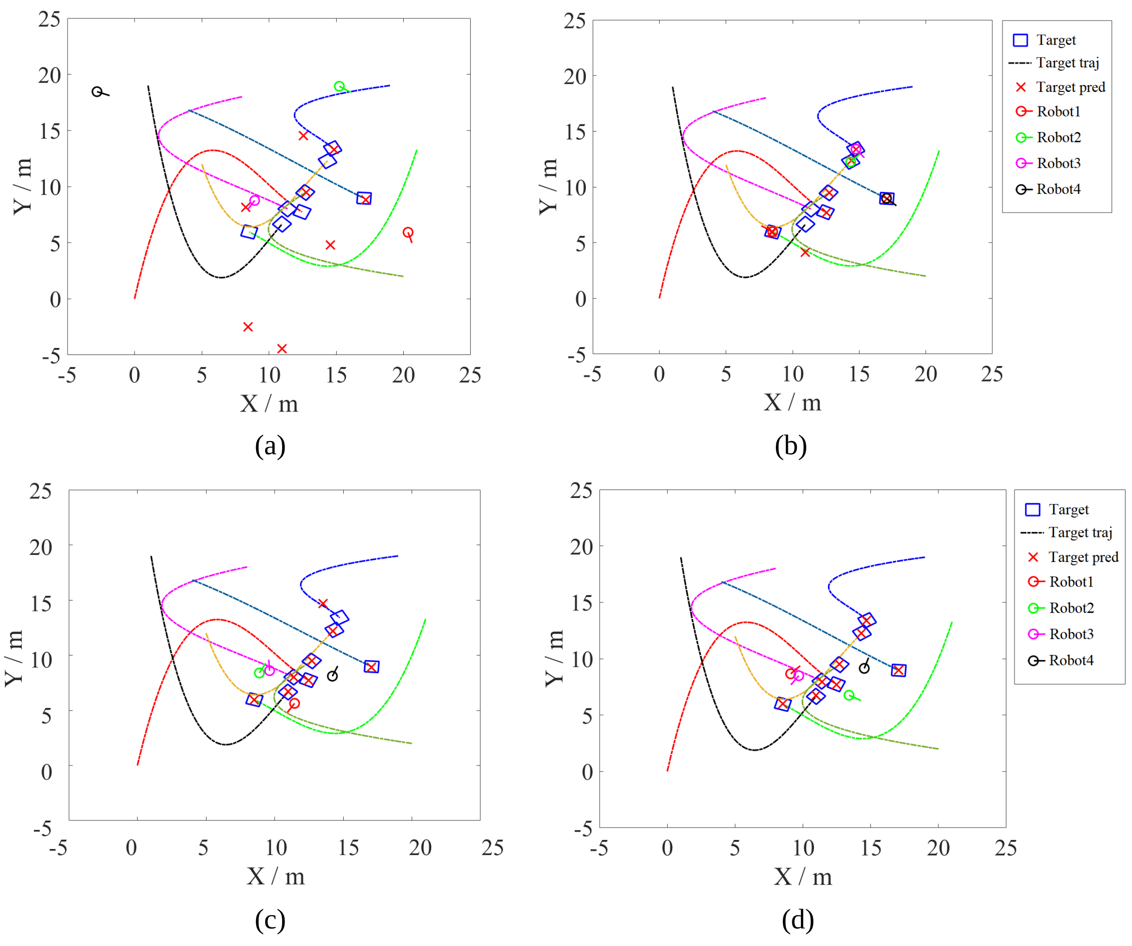


Figure 7.11: Target prediction and tracking performance using (a) random planner, (b) nearest target following planner, (c) RESIN, and (d) centralized GP planner.

sensors, RESIN could reach the same performance with centralized GP, however, that is because the number of sensors is too small and they could only measure only a part of the targets and is not able to measure all the targets, so the average prediction error is relatively small. In order to get the performance we needed, it's best to 4-6 sensors for measuring and tracking the targets.

Also, we then keep the number of sensors constant and change the number of targets. With the workspace and initial setting unchanged. 4 sensors are deployed and the targets number varied from 4 to 8. The result could be seen in

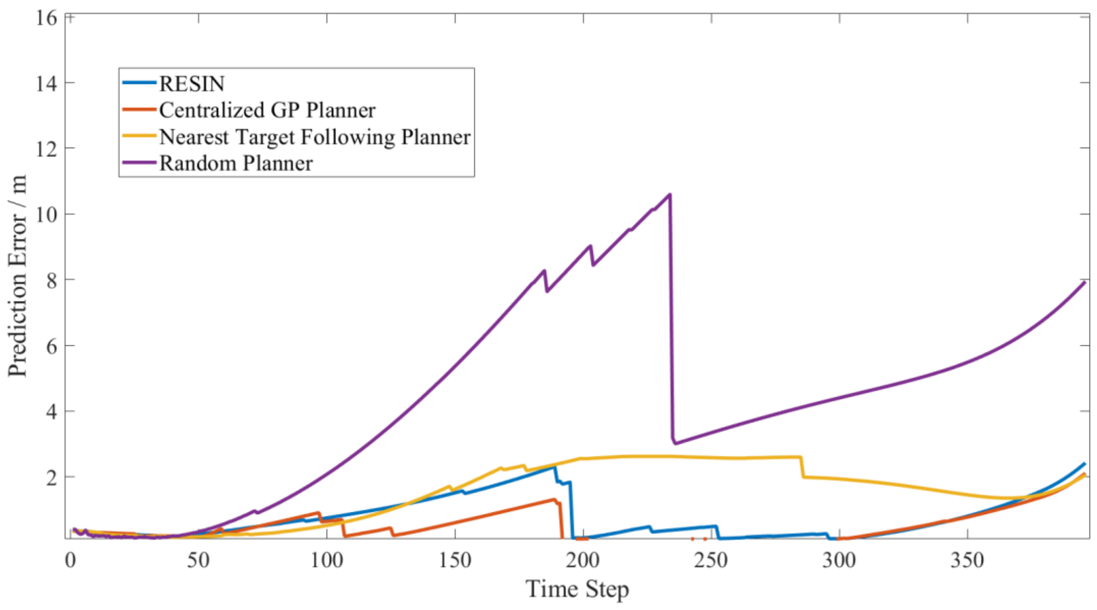


Figure 7.12: Average prediction error of targets using different planning strategies.

figure (7.14) and table (7.2). From the result it can be found that with the increasing of the number of targets, the prediction error is increasing because for each sensor there are more targets needed to predict. Also, the difference of prediction error between RESIN and nearest target following planner is enlarging. Meaning with the increasing of targets, the RESIN shows more applicability in keeping a relatively small error in prediction.

So the conclusion could be reached that compared to other method, RESIN is represented to be more applicable in the situation that the target number is more than sensors number, which is more practical in real world application since the number of sensors could be saved while measuring multiple targets.

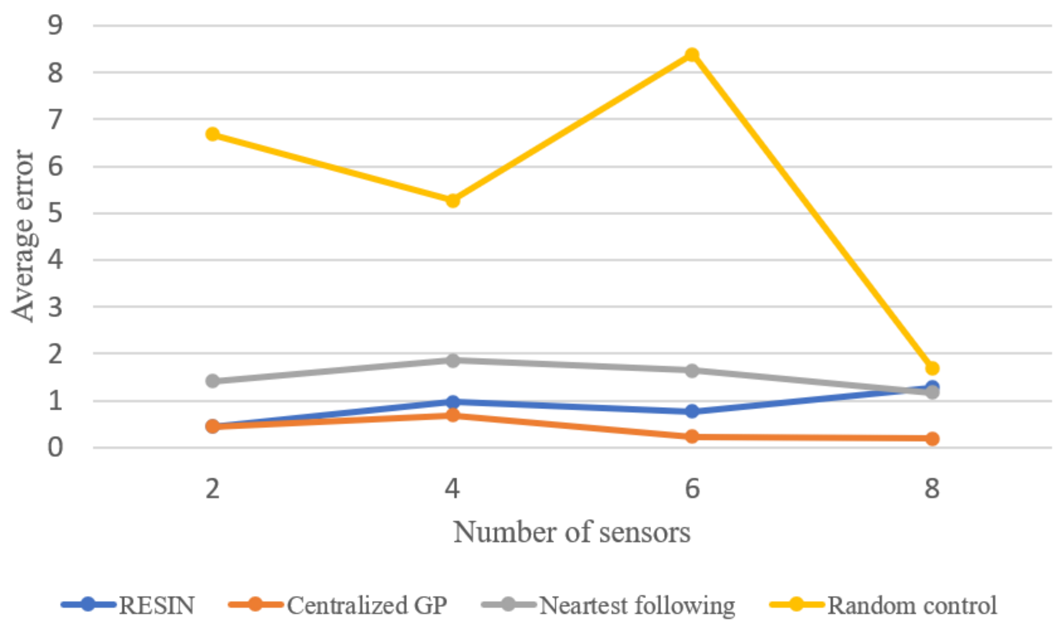


Figure 7.13: Average prediction error of targets using different number of sensors for measuring 8 targets.

Table 7.1: Comparison of the prediction performance using different number of sensors in measuring 8 targets

Number of sensors		2	4	6	8
Average prediction error	RESIN	0.4413	0.9732	0.7691	1.2746
	Centralized GP	0.4413	0.6893	0.2275	0.1902
	Nearest following	1.4083	1.851	1.6356	1.1653
	Random control	6.6845	5.2698	8.3876	1.6969

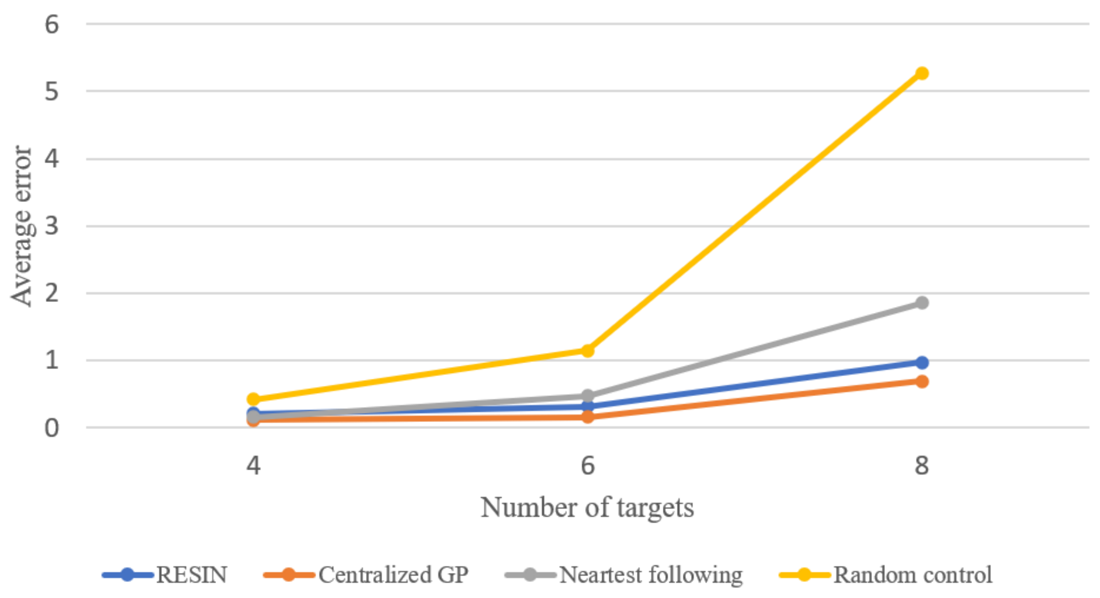


Figure 7.14: Average prediction error of targets using 4 sensors for measuring different number of targets.

Table 7.2: Comparison of the prediction performance using different number of sensors in measuring 8 targets

Number of targets		4	6	8
Average prediction error	RESIN	0.2082	0.3177	0.9732
	Centralized GP	0.1174	0.1586	0.6893
	Nearest following	0.1625	0.4691	1.851
	Random control	0.4184	1.1473	5.2698

CHAPTER 8

CONCLUSION AND FUTURE STEPS

In this thesis, a decentralized GP learning, fusion, and planning (RESIN) algorithm for a sensor network to actively learn the motion pattern of multiple moving targets, and thus planning for each sensor to keep track of all the targets based on the information entropy was proposed. To extract the target's motion from a dense scene flow, an ego-motion extract algorithm was proposed for computing the flow produced by the camera's ego-motion and extract it from the whole dense scene flow. For a better visualization of the scene flow, a 3D reconstruction method was proposed. A decentralized GP fusion method was introduced, which, compared to the ordinary GP, is more robust to rumor propagation and more computational efficient. An information-driven path planning (IPP) method was proposed. Comparing to the traditional target pursuit algorithm, IPP is more information sensitive based on decentralized GP fusion learning and prediction, and therefore presents more proper performance while dealing with multiple targets pursuing problem. Simulations are done to represent the accuracy and efficiency of RESIN and a final simulation was demonstrated to compare RESIN with other method to prove the efficiency of RESIN compared to other traditional method. Also, compared to nearest target following method, RESIN shows more efficiency and applicability in the situation that the sensor number is less than the target number, which means RESIN is more suitable for practical applications.

The future work will further investigate into the data association issue in the decentralized GP fusion, for example, how to associate each keypoint of the target measured in different time frame. Besides, in the simulation, due to the computational cost of scene flow extraction and processing, this thesis only uses the ground truth of the targets' motion, with a gaussian noise $\boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_{\varepsilon})$, where $\boldsymbol{\Sigma}_{\varepsilon} = \epsilon_0^2 \mathbf{I}$ added as the measurement of the targets. In the future work, a more computational efficient motion extraction method should be combined into the simulation. Also, physical experiments using actual mobile robots and sensors for chasing actual targets will be conducted to evaluate this thesis in the physical practical application.

BIBLIOGRAPHY

[1] Nisar R Ahmed and Mark Campbell. Fast consistent chernoff fusion of gaussian mixtures for ad hoc sensor networks. *IEEE transactions on signal processing*, 60(12):6739–6745, 2012.

[2] Nikolay Atanasov, Jerome Le Ny, Kostas Daniilidis, and George J Pappas. Decentralized active information acquisition: Theory and application to multi-robot slam. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4775–4782. IEEE, 2015.

[3] John L Barron and Neil A Thacker. Tutorial: Computing 2d and 3d optical flow. *Imaging Science and Biomedical Engineering Division, Medical School, University of Manchester*, 2005.

[4] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2006.

[5] Mark E Campbell and Nisar R Ahmed. Distributed data fusion: Neighbors, rumors, and the art of collective knowledge. *IEEE Control Systems Magazine*, 36(4):83–109, 2016.

[6] Yanshuai Cao and David J Fleet. Generalized product of experts for automatic and principled fusion of gaussian process predictions. *arXiv preprint arXiv:1410.7827*, 2014.

[7] Philip Dames, Pratap Tokekar, and Vijay Kumar. Detecting, localizing, and tracking an unknown number of moving targets using a team of mobile robots. *The International Journal of Robotics Research*, 36(13-14):1540–1553, 2017.

[8] Marc Deisenroth and Jun Wei Ng. Distributed gaussian processes. In *International Conference on Machine Learning*, pages 1481–1490, 2015.

[9] Silvia Ferrari and Thomas A. Wettergren. *Information-Driven Planning and Control*. MIT Press, 2020.

[10] Silvia Ferrari, Rafael Fierro, and Domagoj Tolic. A geometric optimization approach to tracking maneuvering targets using a heterogeneous mobile sensor network. *Proceedings of the IEEE Conference on Decision and Control*, pages 1080 – 1087, 2010.

[11] Greg Foderaro, Pingping Zhu, Hongchuan Wei, Thomas A Wettergren, and Silvia Ferrari. Distributed optimal control of sensor networks for dynamic target tracking. *IEEE Transactions on Control of Network Systems*, 5(1):142–153, 2018.

[12] Jake Gemerek, Silvia Ferrari, Brian H Wang, and Mark E Campbell. Video-guided camera control for target tracking and following. *IFAC-PapersOnLine*, 51(34):176–183, 2019.

[13] Gabriel M Hoffmann and Claire J Tomlin. Mobile sensor network control using mutual information methods and particle filters. *IEEE Transactions on Automatic Control*, 55(1):32–47, 2010.

[14] Sungmin Hong, Daeyoung Kim, Minkeun Ha, Sungho Bae, Sang Jun Park, Wooyoung Jung, and Jae-Eon Kim. Snail: an ip-based wireless sensor network approach to the internet of things. *IEEE Wireless Communications*, 17(6):34–42, 2010.

[15] J. Joseph, F. Doshi-Velez, A.S. Huang, and et al. A bayesian nonparametric approach to modeling motion patterns. *Auton Robot*, page 383, 2011.

[16] Andreas Krause and Daniel Golovin. Submodular function maximization., 2014.

[17] Andreas Krause and Carlos Guestrin. Near-optimal observation selection using submodular functions. In *AAAI*, volume 7, pages 1650–1654, 2007.

[18] Philip Lenz, Julius Ziegler, Andreas Geiger, and Martin Roser. Sparse scene flow segmentation for moving object detection in urban environments. In *2011 IEEE Intelligent Vehicles Symposium (IV)*, pages 926–932. IEEE, 2011.

[19] Alberto Quattrini Li, Phani Krishna Penumarthi, Jacopo Banfi, Nicola Basilico, Jason M O'Kane, Ioannis Rekleitis, Srihari Nelakuditi, and Francesco Amigoni. Multi-robot online sensing strategies for the construction of communication maps. *Autonomous Robots*, pages 1–21, 2019.

[20] Chang Liu, Yucheng Chen, Jake Gemerek, Hengye Yang, and Silvia Ferrari. Learning recursive bayesian nonparametric modeling of moving targets via mobile decentralized sensors. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8034–8040. IEEE, 2019.

[21] Chang Liu, Shengbo Eben Li, Diange Yang, and J Karl Hedrick. Distributed

bayesian filter using measurement dissemination for multiple unmanned ground vehicles with dynamically changing interaction topologies. *Journal of Dynamic Systems, Measurement, and Control*, 140(3):030903, 2018.

[22] Miao Liu, Girish Chowdhary, Bruno Castra Da Silva, Shih-Yuan Liu, and Jonathan P How. Gaussian processes for learning and control: A tutorial with examples. *IEEE Control Systems Magazine*, 38(5):53–86, 2018.

[23] Moritz Menze and Andreas Geiger. Object scene flow for autonomous vehicles. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3061–3070, 2015.

[24] Mehran Mesbahi and Magnus Egerstedt. *Graph theoretic methods in multiagent networks*. Princeton University Press, 2010.

[25] Frank Nielsen. Chernoff information of exponential families. *arXiv preprint arXiv:1102.2684*, 2011.

[26] Jorge Nocedal and Stephen Wright. *Numerical optimization*. Springer Science & Business Media, 2006.

[27] Ruofei Ouyang, Kian Low, Jie Chen, and Patrick Jaillet. Multi-robot active sensing of non-stationary gaussian process-based environmental phenomena. *13th International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2014*, 1:573–580, 2014.

[28] CE Rasmussen and CKI Williams. *Gaussian Processes for Machine Learning*. Adaptive Computation and Machine Learning. MIT Press, Cambridge, MA, USA, 1 2006.

[29] Amarjeet Singh, Fabio Ramos, Hugh Durrant Whyte, and William J Kaiser. Modeling and decision making in spatio-temporal processes for environmental surveillance. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 5490–5497. IEEE, 2010.

[30] R. Vidal, O. Shakernia, H. J. Kim, and et al. Probabilistic pursuit-evasion games: theory, implementation, and experimental evaluation. *IEEE Transactions on Robotics and Automation*, pages 662–669, 2002.

[31] Hongchuan Wei, Wenjie Lu, Pingping Zhu, Silvia Ferrari, Miao Liu, Robert H Klein, Shayegan Omidshafiei, and Jonathan P How. Information

value in nonparametric dirichlet-process gaussian-process (dpgp) mixture models. *Automatica*, 74:360–368, 2016.

[32] Hongchuan Wei, Pingping Zhu, Miao Liu, Jonathan P How, and Silvia Ferrari. Automatic pan-tilt camera control for learning dirichlet process gaussian process (dpgp) mixture models of multiple moving targets. *IEEE Transactions on Automatic Control*, 2018.

[33] Yunfei Xu, Jongeun Choi, and Songhwai Oh. Mobile sensor network navigation using gaussian processes with truncated observations. *IEEE Transactions on Robotics*, 27(6):1118–1131, 2011.

[34] Koichiro Yamaguchi, David McAllester, and Raquel Urtasun. Robust monocular epipolar flow estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1862–1869, 2013.

[35] Seniha Esen Yuksel, Joseph N Wilson, and Paul D Gader. Twenty years of mixture of experts. *IEEE transactions on neural networks and learning systems*, 23(8):1177–1193, 2012.

[36] Pingping Zhu, Silvia Ferrari, Julian Morelli, Richard Linares, and Bryce Doerr. Scalable gas sensing, mapping, and path planning via decentralized hilbert maps. *Sensors*, 19(7):1524, 2019.