Active SLAM in 3D deformable environments

Mengya Xu[1,2], Liang Zhao[2], Shoudong Huang[2], and Qi Hao[1,3]

Abstract— **This paper considers active SLAM problem for 3D deformable environments where the trajectory of the robot is planned to optimize the SLAM results. A planning strategy combining an efficient global planner with an accurate local planner is proposed to solve the problem. Simulation results under different scenarios have shown that the proposed active SLAM algorithm provides a good balance between accuracy and efficiency as compared to the local planner and the global planner. The MATLAB code of this first active SLAM algorithm for 3D deformable environments is made publicly available[4].**

I. INTRODUCTION

SLAM in deformable environments is an important research topic, due to its wide applications in many different areas such as robotic motion tracking and minimally invasive robotic surgery. A few groups have completed some good work on this challenging topic. When an RGB-D sensor is used to observe the environment, a common approach is to deform the prior or build map directly based on the observations [1] [2] [3]. In some cases, high accuracy is required but the sensor vision is limited, extra techniques are needed to provide additional information. For example, in some surgical cases, computed tomograph (CT) is used to provide an ideal prior model for recovering the deformation [4] [5], while [6] uses GPU and ORB-SLAM to obtain a pose estimation first. More challenging cases when only one monocular camera can be used have also been investigated by many works like [7] [8]. In [9] and [10], the camera motion and the the 3D shape of deformable objects are recovered using an EKF. In [11], some fundamental questions about SLAM in deformable environment are discussed, such as the observability and the consistency.

Planning a good trajectory for the robot/sensor is critical and also challenging in SLAM in deformable environments. A good trajectory of the robot can help obtain high quality SLAM estimate. Intuitively, the robot wants to observe as many features as possible at each time step so that more information can be obtained through the observations. In most of the works of SLAM in deformable environments, the motion of the robot is controlled by human or predetermined. However, it is more desirable if the robot can decide its control action online based on different situations, which is the active SLAM problem in deformable environments to be considered in this paper.

Active SLAM in static environments has been well studied in the last decade. In active SLAM, one important performance criteria is the quality of SLAM estimate. In most cases, the Fisher information matrix or covariance matrix in the corresponding estimation problem is used to build the objective function for selecting the robot motion [12] [13]. The active SLAM problem is often combined with other tasks such as exploration. In these problems, how to balance the performance of other tasks and the quality of the SLAM estimate is a challenge. Different methods have been proposed to consider different tasks. The most frequently used method is to build a utility function to balance different factors [14] [15]. Another typical method is to set thresholds for certain performances [16]. So far, various research groups have been considering different active SLAM problems using different performance metrics, which makes a fair comparison of different research works difficult.

A popular framework for active SLAM is selecting the best action from a finite set of candidate actions [17] [18]. However, computational complexity of evaluating these candidate actions grows exponentially with the size of the action space. In practice, frontier-based exploration [19] [20] is a more widely used approach, where a small subset of locations in the map is selected. However, these approaches can only guarantee convergence to locally optimal policies. Considering the global information, there are works trying to pre-compute and store the information in a special map [21] [22] [23]. However, the pre-calculated map representation is only efficient for static environments with few possible changes. For deformable environments with high dynamics, the information in the map needs to be updated continuously, costing much more time than the traditional active SLAM methods in static environments. Recently, [24] proposed a hierarchical exploration framework, including a local planner to plan detailed paths using dense data and a global planner to plan coarse paths using sparse data. This approach considers the global information, and at the same time significantly improves the computational efficiency. This kind of hierarchical framework is also widely used in multi-robot path planning [25] for exploring partially unknown environment. However, these approaches mainly focus on the map coverage and collision avoidance, leaving the estimation uncertainty during the SLAM process unconsidered.

In this paper, we consider the active SLAM problem in 3D deformable environments which has not been stud-

This work is partially supported by the Science and Technology Innovation Committee of Shenzhen City (No: JCYJ20200109141622964)

Corresponding author: Qi Hao.

[1] Mengya Xu and Qi Hao are with Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen, China. hao.q@sustc.edu.cn

[2] Mengya Xu, Liang Zhao, and Shoudong Huang are with Robotics Institute, University of Technology Sydney, NSW 2008, Australia. Mengya.Xu@student.uts.edu.au

[3]Research Institute of Trustworthy Autonomous Systems, Southern University of Science and Technology, Shenzhen, Guangdong, China.

[4] The MATLAB source codes are made available at https://github.com/MengyaXu/activeSLAM-in-deformable-environments.git

ied yet. The environment we consider is deformable and highly dynamic. The robot needs to take the environment changes into consideration so that it can successfully reach the target region by itself and map the latest environment accurately. There are some works focusing on SLAM in partially dynamic environment, especially in the field of service robot [26] and autonomous driving [27], where features from moving objects need to be distinguished and removed from the SLAM process. However, for SLAM in deformable environments, there is no static part and all the features considered in the SLAM estimate are dynamic. In our previous work [11], some fundamental questions about SLAM in deformable environment have been discussed, including the active SLAM problem. Based on possible assumptions on available information, a greedy active SLAM algorithm is designed and tested in a simple 2D deformable environment. In this paper, we consider 3D point feature based SLAM in deformable environments. As an extension of our previous work, an EKF based active SLAM framework is designed to estimate the SLAM result accurately and efficiently. Different from the previous work that used a greedy planner [11], the planner proposed in this paper is a combination of a global planner and a local planner. Simulation results demonstrate that our proposed framework can get more accurate results than using global planning only, and more efficient than the local greedy planning method.

The paper is organized as follows. Section II reviews models of SLAM in deformable environments based on the reasonable assumptions. Section III presents the details of the proposed active SLAM framework. Experimental results using different simulation scenarios are provided and compared to the global planning and local planning strategies in Section IV. Finally, Section V concludes the paper and presents some future work.

II. EKF SLAM IN DEFORMABLE ENVIRONMENTS

It is well known that the SLAM problem in deformable environments is not solvable unless some assumptions on the possible deformation and/or robot trajectory are made [11]. In this paper, the robot odometry model, observation model and the feature dynamic models are formulated according to the available information considering the 3D case. In this section, we will introduce the EKF based 3D SLAM in deformable environments. It has much less computational cost compared with optimization based approach [11].

In the considered 3D feature based EKF SLAM problem, the state with M features at n-th step is

$$\boldsymbol{\chi}_n = (\mathbf{R}(\Theta_n), \mathbf{x}_n, \mathbf{p}_n^1, \cdots, \mathbf{p}_n^M), \tag{1}$$

where $\mathbf{R}(\Theta_n) \in SO(3)$, $\boldsymbol{x}_n \in \mathbb{R}^3$ and $\mathbf{p}_n^i \in \mathbb{R}^3$ ($i = 1, \cdots, M$) are respectively the robot orientation, robot position, and the coordinate of the i-th feature, all described in the fixed world coordinate frame. Note the features are also changing over time.

The robot odometry model is the same as in SLAM in static environments. In general, for the SLAM problem in deformable environments, some knowledge on both the global rigid motion and the local deformation of the features are available. Therefore, we can assume that the movement of the features includes the global translation $\mathbf{t}$ and the local deformation $\mathbf{d}$, and $\mathbf{r}_t \in SO(3)$ and $\mathbf{r}_d \in SO(3)$ are the corresponding rotation matrices.

Thus, the process model of the SLAM problem is

$$\boldsymbol{\chi}_{n+1} = \boldsymbol{f}(\boldsymbol{\chi}_n, \mathbf{u}_n, \mathbf{w}_n) = \begin{bmatrix} \mathbf{R}(\Theta_n + \Omega_n + \mathbf{w}_n^{\Omega}) \\ \mathbf{x}_n + \mathbf{R}(\Theta_n)\mathbf{v}_n + \mathbf{w}_n^{v} \\ \mathbf{p}_n^1 + \mathbf{r}_t\mathbf{t}_n + \mathbf{r}_d\mathbf{d}_n^1 + \mathbf{w}_n^1 \\ \vdots \\ \mathbf{p}_n^M + \mathbf{r}_t\mathbf{t}_n + \mathbf{r}_d\mathbf{d}_n^M + \mathbf{w}_n^M \end{bmatrix}, \tag{2}$$

where $\boldsymbol{\chi}_n \sim \mathcal{N}(\hat{\boldsymbol{\chi}}_n, \mathbf{P}_n)$, $\boldsymbol{u}_n = [\Omega_n, \mathbf{v}_n]$ is the control input, and $\mathbf{w}_n = \left[\mathbf{w}_n^{\Omega}, \mathbf{w}_n^{v}, \mathbf{w}_n^{j=1:M}\right] \sim \mathcal{N}(\mathbf{0}, \boldsymbol{Q}_n)$ is the noise.

Note that the feature movement noise $\mathbf{w}_n^j$ contains two parts, the global translation noise $\mathbf{w}_{n,j}^t$ and the local deformation $\mathbf{w}_{n,j}^d$. That is, $\mathbf{w}_n^j = \mathbf{w}_{n,j}^t + \mathbf{w}_{n,j}^d$.

The observation model also contains two parts, the measurement of the features and the measurement of the structure.

$$\mathbf{Z}_n = [\mathbf{z}_n^f; \mathbf{z}_n^c] = \boldsymbol{h}_n(\boldsymbol{\chi}_n, \boldsymbol{\epsilon}_n). \tag{3}$$

Feature measurement $\mathbf{z}_{n,j}^f$ is the observations from the robot sensor to the j-th features, just as in static environment.

Structure measurement $\mathbf{z}_{n,j_1,j_2}^c$ are the observations of the constraints between each pair of features, $\mathbf{p}^{j_1=1:M}$ and $\mathbf{p}^{j_2=1:M}$. In our experiment, the structure measurement between feature $\mathbf{p}^{j_1}$ and $\mathbf{p}^{j_2}$ is set to be the relative position between them:

$$\mathbf{z}_{n,j1,j2}^c = \mathbf{f}^{loc}(n, j_2) - \mathbf{f}^{loc}(n, j_1) + \boldsymbol{\epsilon}_n^c, \tag{4}$$

where $\mathbf{f}^{loc}(n, j)$ is the feature's position relative to $\mathbf{p}_n^1$ in the local coordinate, given by

$$\mathbf{f}^{loc}(n, j) = \mathbf{p}_n^j - \mathbf{p}_n^1. \tag{5}$$

The observation noise $\boldsymbol{\epsilon}_n = [\boldsymbol{\epsilon}_n^f, \boldsymbol{\epsilon}_n^c] \sim \mathcal{N}(\mathbf{0}, \boldsymbol{O}_n)$, where $\boldsymbol{\epsilon}_n^f$ and $\boldsymbol{\epsilon}_n^c$ corresponds to the feature measurement noise and the structure measurement noise respectively.

The EKF based SLAM algorithm in deformable environments is presented in Algorithm 1.

III. ACTIVE SLAM IN DEFORMABLE ENVIRONMENTS

In this section, we first introduce the traditional planning method that minimizes a certain criterion to get the action in each step, which we call local planner here. Secondly, the global planner that is similar to [24] but for deformable environments is presented. Finally, we propose a combined planner that combines the local planning method and the global planning method.

A. The active SLAM problem

For the active SLAM problem considered in this paper, given a deformable object of which the global translation model and the local deformation model are known, we assume there are a known number of features distributed on the surface of the object, and some prior knowledge of the

feature distribution is known. The robot starts from a fixed location in the environment. The objective is to plan the robot trajectory for a given time horizon, so that it can observe the object completely, and estimate the observed features and the robot poses accurately.

Algorithm 1 EKF SLAM in deformable environment

Input: $\hat{\boldsymbol{\chi}}_n$, $\mathbf{P}_n$, $\mathbf{u}_n$, $\mathbf{Z}_{n+1}$
Output: $\hat{\boldsymbol{\chi}}_{n+1}$, $\mathbf{P}_{n+1}$
Propagation:
$\hat{\boldsymbol{\chi}}_{n+1|n} \leftarrow \boldsymbol{f}(\hat{\boldsymbol{\chi}}_n, \mathbf{u}_n, \mathbf{0})$, $\mathbf{P}_{n+1|n} \leftarrow \boldsymbol{F}_n \mathbf{P}_n \boldsymbol{F}_n^T + \boldsymbol{G}_n \boldsymbol{Q}_n \boldsymbol{G}_n^T$
Update:
$\boldsymbol{S}_{n+1} \leftarrow \boldsymbol{H}_{n+1} \mathbf{P}_{n+1|n} \boldsymbol{H}_{n+1}^T + \boldsymbol{O}_{n+1}$
$\boldsymbol{K}_{n+1} \leftarrow \mathbf{P}_{n+1|n} \boldsymbol{H}_{n+1}^T \mathbf{S}_{n+1}^{-1}$
$\boldsymbol{y}_{n+1} \leftarrow \mathbf{Z}_{n+1} - \boldsymbol{h}_{n+1}(\hat{\boldsymbol{\chi}}_{n+1|n}, \mathbf{0})$
$\hat{\boldsymbol{\chi}}_{n+1} \leftarrow \hat{\boldsymbol{\chi}}_{n+1|n} + \boldsymbol{K}_{n+1} \boldsymbol{y}_{n+1}$
$\mathbf{P}_{n+1} \leftarrow (\boldsymbol{I} - \boldsymbol{K}_{n+1} \boldsymbol{H}_{n+1}) \mathbf{P}_{n+1|n}$

B. Local planner

The local planner is the greedy method, where the information that will be obtained in the next step is maximized to obtain the robot control action. The information gained in terms of the SLAM estimate can be described by the resulting covariance matrix after the control action is taken and the information from the new observations are used.

Concretely, given $\hat{\boldsymbol{\chi}}_n$ and $\mathbf{P}_n$, we would like to select the control vector $\boldsymbol{u}_n$ such that a certain metric (e.g. the trace) of the covariance matrix in the next step is optimized. That is, the trace($\mathbf{P}_{n+1}$) is expected to be as small as possible:

$$obj = \min \text{trace}(\mathbf{P}_{n+1}), \tag{6}$$

where $\mathbf{P}_{n+1}$ is obtained by Algorithm 1.

Note that the feature observation $\mathbf{Z}_{n+1}$ is not available when the planning is performed, so we assume that no new feature will be observed and the estimation will not be updated after the observation (zero-innovation) [28].

$$\begin{aligned} &\mathbf{Z}_{n+1} - \boldsymbol{h}_{n+1}(\hat{\boldsymbol{\chi}}_{n+1|n}) = 0, \\ &\hat{\boldsymbol{\chi}}_{n+1} = \hat{\boldsymbol{\chi}}_{n+1|n}. \end{aligned} \tag{7}$$

C. Global planner

In the global planning, the objective is to generate a set of viewpoints, so that the robot can observe each feature at least once by visiting the viewpoints. Here, a viewpoint $\mathbf{g}$ is a 3D position. The reward of each viewpoint, r, is defined as the number of unobserved features that can be observed at $\mathbf{g}$. In our proposed method, the global planner is a coarse but efficient planner. To simplify the calculation process, we do not consider the orientation of the robot, so features that within the distance of the sensor range are considered to be able to be observed. Note that the reward of each candidate viewpoint is updated according to the previous selected viewpoints. As the same feature can be observed from multiple viewpoints, once it is observed by the selected viewpoint, it needs to be removed from others' field-of-view, and thus the reward needs to be updated accordingly.

Algorithm 2 presents the process of viewpoint set generation, which is similar to the sampling process shown in [24]. The algorithm first generates a set of viewpoint candidates $\mathcal{G}_{cand}$ distributed uniformly in the 3D space around the object. Secondly, the rewards of all viewpoint candidates in $\mathcal{G}_{cand}$ are computed based on the initial robot pose and the initially estimated feature positions. The reward of each candidate viewpoint is the number of features that can be observed in the viewpoint, minus the number of the features that have been observed by the robot in the initial pose.

Algorithm 2 Viewpoint set generation algorithm

Input: traversable space $\mathcal{S}$
Output: viewpoint set $\mathcal{G}_{final}$

```
Generate a set of viewpoint candidates G_cand in S
For each candidate g_i, calculate the number of the unobserved features that can be observed at g_i, and set it to be its reward r_i
c_best = ∞
for i = 1 : K do
    G = ∅
    G'_cand = G_cand
    R = Σ_{i=1}^{length(G'_cand)} r_i
    while R ≠ 0 do
        Probabilistically pick viewpoint g' from G'_cand
        Remove g' from G'_cand
        G ← G ∪ g'
        Update r_i for all viewpoints in G'_cand
        R = Σ_{i=1}^{length(G'_cand)} r_i
    end while
    Compute cost c using equation (8)
    if c < c_best then
        G_final = G, c_best = c
    end if
end for
return G_final
```

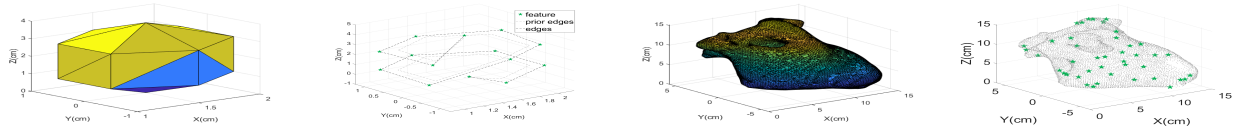
Next, a process is iterated K times to determine the final viewpoint set $\mathcal{G}_{final}$ that contains the viewpoints to be visited in order. Here, K is the number of sample sets to be compared, which can be determined according to the experiment requirement. In each iteration, a subset of viewpoints $\mathcal{G}$ is generated from $\mathcal{G}_{cand}$ according to their rewards. Concretely, the viewpoints are selected with probabilities proportional to their rewards and put in $\mathcal{G}$. After a viewpoint is selected, the rewards of the remaining viewpoints are reduced accordingly. Each iteration process finishes when the total reward of the remaining candidate viewpoints is zero.

After the sampling process, we obtain K sets of viewpoints. The one with the minimum cost function will be selected to be the final viewpoint set $\mathcal{G}_{final}$. The cost function is defined as

$$c = \sum_{i=1}^{I} d_{i,i+1}, \tag{8}$$

where $d_{i,i+1}$ is the Euclidean distance between two adjacent viewpoints $\mathbf{g}_i$ and $\mathbf{g}_{i+1}$ in $\mathcal{G}$.

After the viewpoint set is determined, the viewpoint $\mathbf{g}_i$ is set to be the current goal point $\mathbf{g}_{cur}$ one by one from



(a) Mesh model of the polygon environment

(b) Feature model of the polygon environment

(c) Mesh model of the heart environment

(d) Feature model of the heart environment

Fig. 1: Environment models.

$i = 1 : N_g$ in the order of the set, where N_g is the number of viewpoints in the set. In each step, the robot selects the action that minimizes the distance D between the robot position $\mathbf{x}_{n+1|n}$ and the position of the current goal point $\mathbf{g}_{cur}$:

$$obj = \min D(\mathbf{x}_{n+1|n}, \mathbf{g}_{cur}), \tag{9}$$

Once all the viewpoints are visited, the next viewpoint will be the first viewpoint of the set. That is, if $i = N_g$, $i+1 = 1$.

D. Combined planner

The combined planner is a combination of the local planner and the global planner, aiming to balance the performance in accuracy, coverage and processing time. We use the global planner to generate the viewpoint set to ensure that all features can be observed as soon as possible. At the same time, the local planner which aims to minimize the uncertainty will be used to improve the accuracy.

In particular, the robot action $\mathbf{u}_n$ is selected from a set of candidate actions $\mathbf{U} = \{\mathbf{u}_n^i | i = 1, 2, ..., N_u\}$ in each step, where N_u is the number of the candidate actions. The robot pose after taking each candidate action is predicted, and the distance D between the predicted robot pose and the current goal point is calculated. The action set is sorted according to the distances from small to large, and then the first n_u candidate actions are selected to be the new action set. This process ensures that all actions in the new action set are likely to guide the robot to approach the goal point. The final action will be selected from the new action set using the local planner, which minimizes the trace of the covariance matrix, as shown in eq. (6). Here, n_u is the number of the actions in the new action set, where smaller value of n_u means larger probability of the robot to approach the goal point. If $n_u = 1$, it becomes the global planner. On the contrary, if $n_u = N_u$, it is the same as the local planner.

IV. Simulation Results

A. Models and simulation settings

The process model and feature movement models for the active SLAM are described in Section II. As stated above, the movement of the features includes the global translation $\mathbf{t}$ and the local deformation $\mathbf{d}$. Here, we consider the linear case only, that is, $\mathbf{r}_t = \mathbf{I}_3$ and $\mathbf{r}_d = \mathbf{I}_3$.

For the feature measurement model, the j-th feature observed at the n-th step is given by

$$\mathbf{z}_{n,j}^f = \mathbf{R}(\Theta_n)(\mathbf{x}_n - \mathbf{p}_n^j) + \boldsymbol{\epsilon}_n^f. \tag{10}$$

Two environments with different target deformable objects are used to validate the algorithms in simulation, where the target object is what we want to estimate. One is a polygon environment that contains a created simple object model, as shown in Fig. 1(a) and 1(b). The other is a heart environment that contains a heart model segmented from a CT scan, downloaded from OpenHELP [29], as shown in Fig. 1(c) and 1(d). Our objective is to observe the target object and obtain accurate SLAM estimate.

For the polygon model, we set 14 point features to form a prismatic impenetrable object, as shown in Fig. 1(b). Globally, the object moves back and forth along the x-axis regularly by small degrees, that is, all features move forward 0.1 cm and then come back in the next step. Locally, the deformation is a regular expansion and shrinkage. In our simulation, we set the first feature to be the anchor point. All the other features move away from the anchor point 0.2 cm in each time step for 3 steps, and then move back 3 steps in the same distance. The total step of the robot motion is set to be 100. The initial robot pose is $[0, 0, 0, 0, 0, 0]$, and the sensor range is 3 cm. The covariance matrix of control noise $[\mathbf{w}_n^\Omega, \mathbf{w}_n^v]$ is $\mathrm{diag}[(0.02\mathrm{rad})^2, (0.02\mathrm{rad})^2, (0.02\mathrm{rad})^2, (0.03\mathrm{cm})^2, (0.03\mathrm{cm})^2, (0.03\mathrm{cm})^2]$. The covariance matrix of feature movement noise $\mathbf{w}_n^j = \mathbf{w}_{n,j}^t + \mathbf{w}_{n,j}^d$ is $\mathrm{diag}[(0.03\mathrm{cm})^2 + (0.05\mathrm{cm})^2, (0.03\mathrm{cm})^2 + (0.05\mathrm{cm})^2, (0.03\mathrm{cm})^2 + (0.05\mathrm{cm})^2]$. And the covariance matrix of observation noise $\boldsymbol{\epsilon}_n^j$ also contains two parts. For the feature measurement noise $\boldsymbol{\epsilon}_n^f$, the corresponding covariance matrix is $\mathrm{diag}[(0.02\mathrm{cm})^2, (0.02\mathrm{cm})^2, (0.02\mathrm{cm})^2]$. For the structure measurement noise $\boldsymbol{\epsilon}_n^c$, the corresponding covariance matrix is $\mathrm{diag}[(0.5\mathrm{cm})^2, (0.5\mathrm{cm})^2, (0.5\mathrm{cm})^2]$. Note that the deformable objects are impenetrable, so in the planning algorithms, we restrict the robot to keep a safe distance of 0.3 cm from the object surface.

For the heart model, it was segmented from a CT scan of a healthy, young male undergoing shock room diagnostics. There are thousands of vertex that form the mesh model, as shown in Fig. 1(c). In the experiment, we randomly pick fifty of them to be the features that are used in the decision making process and the SLAM estimation process, as the green stars show in Fig. 1(d). The models and parameters used in the heart model are the same as those in the polygon model, except that the sensor range is set to be 10 cm, and the safe distance between the robot and the object surface is 3 cm. The total step of the robot is set to be 200.

B. Simulation results

The coverage is compared by counting the total number of the observed feature times[1] in the whole time horizon,

[1]If 3 features are observed 2 time each, then the feature times is 6.

and the time steps that all features have been observed at least once. The accuracy is measured by calculating the maximum/average error between the estimated values and the ground truth, including the robot pose error and the feature position error. We record the processing time of the decision making part to compare the computational cost of determining the next control. For each algorithm, we perform the simulation 5 times. The corresponding figures shown in this section are the representative ones.

1) Polygon environment: We first present the SLAM results in the polygon environment. The ground truth of the robot trajectory and the features and the results based on different methods (including the estimated poses and the estimated features, and the covariance ellipse of the features) are shown in Fig. 2.

TABLE I: Coverage for the polygon environment

	No.	*Pre*	*Local*	*Global*	*Combined*
Observed feature times [1]	1	354	681	289	583
	2	393	687	257	515
	3	359	750	266	505
	4	338	655	324	586
	5	292	702	253	518
Steps [2]	1	35	– [3]	19	24
	2	24	–	26	24
	3	24	–	22	29
	4	25	–	22	24
	5	24	–	19	22

[1] The total number of feature times that observed in the whole time horizon.
[2] The time steps that all features have been observed at least once.
[3] '–' means the robot does not observe all the features within the time horizon.

Coverage. In this part, we show the coverage performance of the different methods. The performance of the coverage task is evaluated by counting the total number of the observed feature times in the whole time horizon and the time steps needed to observe all the features at least once, as Table I shows. Four methods are performed, including using a predetermined path, the global planner, the local planner and using the proposed combined planner.

When the predetermined path is used, although all features can be observed at least once eventually, it takes at most 35 steps. What's more, as the path is predetermined without considering the movement of the object, it has the risk of collision. Although this can be solved by setting the path radius larger, the premise is that there are enough free space around the object, which is impractical in many cases. However, this can be easily solved by using active SLAM algorithms by restricting the motion of the robot.

The number of observed features by using the local planner is the largest. However, it cannot observe all features within the time horizon. This is because the local planner only consider the information gain to improve the accuracy, causing the robot continuously re-visiting the previously observed features to reduce the estimation error, as shown in Fig. 2(b).

The global planner costs the least steps (i.e. 19 steps) to observe all features at least once in this experiment. However, the total number of the observed feature times in the whole process is the least. The reason is that when we select the viewpoints, we consider the largest number of features that can be observed at that viewpoint, and in the planning process, we only consider to minimize the distance between the robot and the goal point. It is possible that the total number of the observed feature times is small, because the robot may not observe all features that are expected to be observed at the selected viewpoints, especially when the environment is deforming. For the same reason, it is possible that the steps needed for observing all features are large.

For the proposed combined planner, its performance is as expected. More features can be observed during the whole process compared with the global planner and using a predetermined path. The number of steps used to observe all features is much smaller when compared with the predetermined path and the local planner.

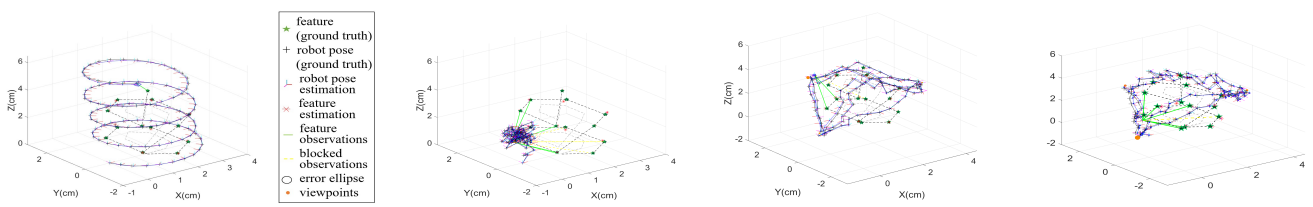
Accuracy. In this part, we compare the accuracy performance of the obtained active SLAM results using different methods. The results of the pose error based on different methods in a single run are shown in Fig. 4(a).

The robot pose error at the n-th step is calculated by:

$$e_x = \sqrt{\mathbf{e}_\theta^{\mathrm{T}}\mathbf{e}_\theta + \mathbf{e}_x^{\mathrm{T}}\mathbf{e}_x}, \tag{11}$$

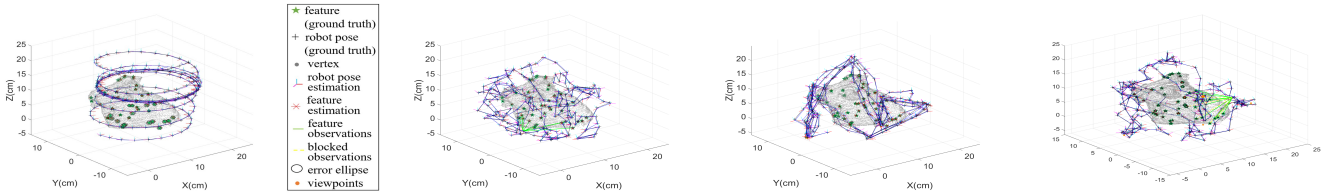
where $\mathbf{e}_\theta = \Theta_n - \hat{\Theta}_n$ is the error between the orientation ground truth and the estimated orientation, and $\mathbf{e}_x = \mathbf{x}_n - \hat{\mathbf{x}}_n$ is the error between the position ground truth and the estimated position. The combined planner and the local planner perform almost the same in terms of the pose error. The global planning algorithm and using the predetermined path obtain relatively larger errors. The average robot pose error and maximum robot pose error of all runs are shown in Table II. It suggests that the combined planner can obtain much smaller errors than the global planner or using a predetermined path. Compared with the local planner, the combined planner is roughly the same.

Besides the robot pose error, Table II also shows the maximum and average feature estimation errors in the last step. The error of the j-th feature can be obtained by calculating



(a) Predetermined (b) Local (c) Global (d) Combined

Fig. 2: Result of using different active SLAM methods in the polygon environment.



(a) Predetermined (b) Local (c) Global (d) Combined

Fig. 3: Results of using different active SLAM methods in the heart environment.

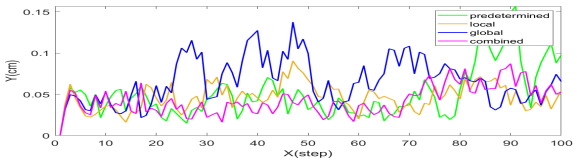
TABLE II: Estimation error for the polygon environment

	Active SLAM			
	Pre	***Local***	***Global***	***Combined***
Max error robot (cm)	0.7433	0.240	0.2750	0.2258
Ave error robot (cm)	0.1878	0.0810	0.1219	0.0925
Max error feature (cm)	0.3316	0.3477	0.3434	0.2622
Ave error feature (cm)	0.1799	0.1199	0.1421	0.1109

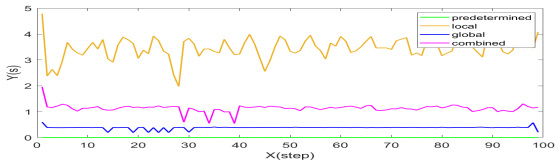
the Euclidean distance between the feature position ground truth and the estimated feature position.

$$e_p = \sqrt{(\mathbf{p}^j - \hat{\mathbf{p}}^j)^{\mathrm{T}}(\mathbf{p}^j - \hat{\mathbf{p}}^j)}, \tag{12}$$

Obviously, the combined planner has advantages over the local planner. Compared with the global planner or using a predetermined path, the combined one can get much smaller maximum error and average error.



(a) Pose error



(b) Decision making time

Fig. 4: Comparison of using different active SLAM methods in the polygon environment.

Processing time. For efficiency, we compare the processing time in the decision making part, that is, the time used to calculate the next action of the robot. Fig. 4(b) shows the results of a single run. We can see that using a predetermined path costs the least time, since there is no decision to make. The combined method costs about 2.5 times longer than the global planning method. The local planner costs much longer time than the others. What's more, the decision making time cost by the local planner is related to the number of the features, because we need to predict the visibility of all features to calculate the covariance matrix.

2) Heart environment: Further tests were run for the heart environment. The same four strategies are compared, and the results are shown in Fig. 3.

Coverage. The coverage performance in terms of the observed feature times and the steps to observe all features are shown in Table III. We can see great advantage of using the combined planner over using a predetermined path and the global method. The combined planner uses much fewer steps to observe all features, and during the whole time horizon, the observed feature times is much more than those two. Although the local planner has more observed feature times in the whole time horizon, it costs more steps to observe all features at least once.

TABLE III: Coverage for the heart environment

	No.	***Pre***	***Local***	***Global***	***Combined***
Observed feature times	1	1358	2042	864	1619
	2	1181	2098	1233	1597
	3	1579	2134	1089	1515
	4	1216	2019	1097	1594
	5	844	2108	922	1664
Steps	1	66	31	72	25
	2	71	62	63	35
	3	70	51	48	35
	4	65	75	47	35
	5	65	74	30	30

TABLE IV: Estimation error for the heart environment

	Pre	***Local***	***Global***	***Combined***
Max error robot (cm)	11.0275	0.4094	0.6529	0.5740
Ave error robot (cm)	0.6714	0.0998	0.1582	0.1203
Max error feature (cm)	0.4554	0.3384	0.2892	0.1975
Ave error feature (cm)	0.1670	0.1025	0.1186	0.1047

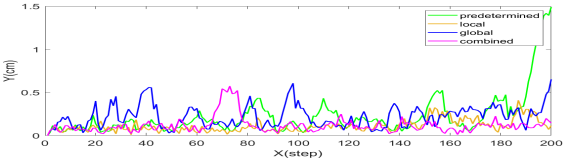
Accuracy. The robot pose errors based on different methods in a single run are shown in Fig. 5(a). It is obvious that the combined method gets relatively smaller errors than using a predetermined path and the global method. The detailed values about the maximum and average errors of 5 runs are shown in Table IV. It also suggests that the combined planner can get much accurate results compared with those two methods. The errors obtained by the local planner and the combined planner are close, and much smaller than those obtained by the other two methods.

Processing time. The performance in processing time is as expected. The result of a single run is shown in Fig. 5(b). The combined planner cost about 2 times longer than the global planner.

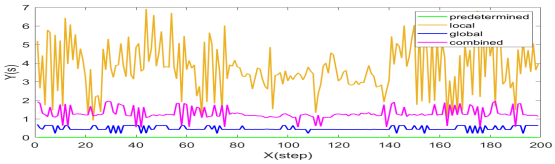
V. Conclusion and Future Work

This paper proposes an active SLAM algorithm for 3D deformable environments. Because of the combination of the efficient global planner and the accurate local planner, the proposed algorithm shows better performance in accuracy as compared with the global planner. It is also demonstrated that the proposed algorithm has satisfactory performance in accuracy and much lower computational cost as compared with the local planner.

This research is the first step in developing an efficient active SLAM algorithm in deformable environments. In the



(a) Pose error



(b) Decision making time

Fig. 5: Comparison of using different active SLAM methods in the heart environment.

future, we will validate our approach in more practical environments with more complex feature movements and robot motion constraints. Some other planning strategies will be investigated and evaluated. We will also consider to conduct real-time active SLAM implementation in the future.

REFERENCES

[1] Richard A. Newcombe, Dieter Fox, and Steven M. Seitz. Dynamicfusion: Reconstruction and tracking of non-rigid scenes in real-time. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 343–352, 2015.

[2] Wei Gao and Russ Tedrake. Surfelwarp: Efficient non-volumetric single view dynamic reconstruction. In *Robotics: Science and System (RSS)*, 06 2018.

[3] Matthias Innmann, Michael Zollhöfer, Matthias Nießner, Christian Theobalt, and Marc Stamminger. Volumedeform: Real-time volumetric non-rigid reconstruction. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision – ECCV 2016*, pages 362–379, Cham, 2016. Springer International Publishing.

[4] Mingsong Dou, Jonathan Taylor, Henry Fuchs, Andrew Fitzgibbon, and Shahram Izadi. 3d scanning deformable objects with a single rgbd sensor. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 493–501, 2015.

[5] Jingwei Song, Jun Wang, Liang Zhao, Shoudong Huang, and Gamini Dissanayake. Dynamic reconstruction of deformable soft-tissue with stereo scope in minimal invasive surgery. *IEEE Robotics and Automation Letters*, 3(1):155–162, 2018.

[6] Jingwei Song, Jun Wang, Liang Zhao, Shoudong Huang, and Gamini Dissanayake. Mis-slam: Real-time large-scale dense deformable slam system in minimal invasive surgery based on heterogeneous computing. *IEEE Robotics and Automation Letters*, 3(4):4068–4075, 2018.

[7] Jose Lamarca, Shaifali Parashar, Adrien Bartoli, and J. M. M. Montiel. Defslam: Tracking and mapping of deforming scenes from monocular sequences. *IEEE Transactions on Robotics*, PP:1–13, 09 2020.

[8] Antonio Agudo. Total estimation from rgb video: On-line camera self-calibration, non-rigid shape and motion. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 8140–8147, 2021.

[9] Antonio Agudo, Francesc Moreno-Noguer, Begoña Calvo, and J. Montiel. Sequential non-rigid structure from motion using physical priors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38:979 – 994, 05 2016.

[10] Antonio Agudo, Francesc Moreno-Noguer, Begoña Calvo, and J. Montiel. Real-time 3d reconstruction of non-rigid shapes with a single moving camera. *Computer Vision and Image Understanding*, 153:37–54, 05 2016.

[11] Shoudong Huang, Yongbo Chen, Liang Zhao, Yanhao Zhang, and Mengya Xu. Some research questions for slam in deformable environments. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2021)*, 2021.

[12] Shoudong Huang, N.M. Kwok, G. Dissanayake, Q.P. Ha, and Gu Fang. Multi-step look-ahead trajectory planning in slam: Possibility and necessity. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 1091–1096, 2005.

[13] Vadim Indelman, Luca Carlone, and Frank Dellaert. Planning under uncertainty in the continuous domain: A generalized belief space approach. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6763–6770, 2014.

[14] Alexei Makarenko, Stefan Williams, Frédéric Bourgault, and Hugh Durrant-Whyte. An experiment in integrated exploration. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 1, pages 534–539 vol.1, 2002.

[15] Henry Carrillo, Philip Dames, Vijay Kumar, and Jose Castellanos. Autonomous robotic exploration using a utility function based on rényi's general theory of entropy. *Autonomous Robots*, 42, 02 2018.

[16] Cindy Leung, Shoudong Huang, and Gamini Dissanayake. Active slam using model predictive control and attractor based exploration. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5031, 2006.

[17] Jose Luis Blanco, J.-A Fernández-Madrigal, and Javier González-Jiménez. A novel measure of uncertainty for mobile robot slam with rao blackwellized particle filters. *I. J. Robotic Res.*, 27:73–89, 01 2008.

[18] Luca Carlone, Jingjing Du, Miguel Kaouk, Basilio Bona, and Marina Indri. Active slam and exploration with particle filters using kullback-leibler divergence. *Journal of Intelligent and Robotic Systems*, 75, 08 2013.

[19] Matan Keidar and Gal A. Kaminka. Efficient frontier detection for robot exploration. *Int. J. Rob. Res.*, 33(2):215–236, feb 2014.

[20] Brian Yamauchi. A frontier-based approach for autonomous exploration. In *Proceedings 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation CIRA'97. 'Towards New Computational Principles for Robotics and Automation'*, pages 146–151, 1997.

[21] Zichao Zhang and Davide Scaramuzza. Fisher information field: an efficient and differentiable map for perception-aware planning. *ArXiv*, abs/2008.03324, 2020.

[22] Brian Ichter, Benoit Landry, Edward Schmerling, and Marco Pavone. Robust motion planning via perception-aware multiobjective search on gpus. *Proc.Int. Symp. Robot. Research (ISRR)*, 05 2017.

[23] Vadim Indelman. No correlations involved: Decision making under uncertainty in a conservative sparse information space. *IEEE Robotics and Automation Letters*, 1:1–1, 01 2016.

[24] Chao Cao, Hongbiao Zhu, Howie Choset, and Ji Zhang. TARE: A Hierarchical Framework for Efficiently Exploring Complex 3D Environments. In *Proceedings of Robotics: Science and Systems*, Virtual, July 2021.

[25] Luca Bartolomei, Marco Karrer, and Margarita Chli. Multi-robot coordination with agent-server architecture for autonomous navigation in partially unknown environments. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1516–1522, 2020.

[26] Su-Yong An, Lae-Kyoung Lee, and Se-Young Oh. Ceiling vision-based active slam framework for dynamic and wide-open environments. In *Autonomous Robots*, page 40:291–324, 2015.

[27] Nikolas Brasch, Aljaz Bozic, Joe Lallemand, and Federico Tombari. Semantic monocular slam for highly dynamic environments. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 393–400, 2018.

[28] Yongbo Chen, Shoudong Huang, Robert Fitch, and Jianqiao Yu. Efficient active slam based on submap joining, graph topology and convex optimization. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5159–5166, 2018.

[29] Hannes Kenngott, J Wünscher, Martin Wagner, A Preukschas, annalaura Wekerle, Peter Neher, Stefan Suwelack, Stefanie Speidel, Felix Nickel, Dare Oladokun, Lorenzo Albala, Lena Maier-Hein, Rüdiger Dillmann, Hans-Peter Meinzer, and Beat Müller. Openhelp (heidelberg laparoscopy phantom): development of an open-source surgical evaluation and training tool. *Surgical endoscopy*, 29(11):3338—3347, November 2015.