# Active and Interactive Mapping with Dynamic Gaussian Process Implicit Surfaces for Mobile Manipulators

Liyang Liu[1], Simon Fryc[1], Lan Wu[1], Thanh Vu[1], Gavin Paul[1] and Teresa Vidal-Calleja[1]

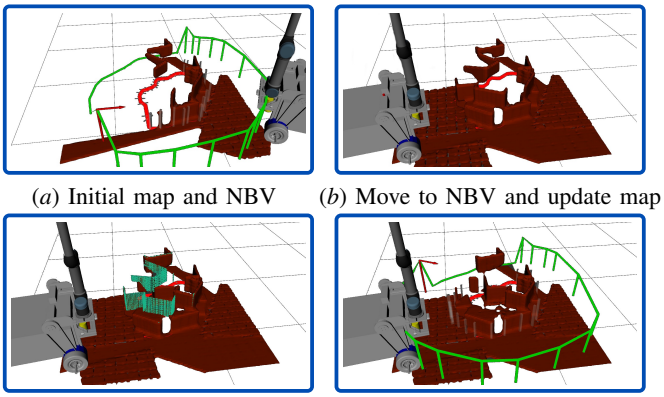
***Abstract*— In this letter, we present an interactive probabilistic mapping framework for a mobile manipulator picking objects from a pile. The aim is to map the scene, actively decide where to go next and which object to pick, make changes to the scene by picking the chosen object, and then map these changes alongside. The proposed framework uses a novel dynamic Gaussian Process (GP) Implicit Surface method to incrementally build and update the scene map that reflects environment changes. Actively the framework provides the next-best-view, balancing the need for picking object reachability with map information gain (IG). To enforce a priority of visiting boundary segments over unknown regions, the IG formulation includes an uncertainty gradient-based frontier score by exploiting the GP kernel derivative. This leads to an efficient strategy that addresses the often conflicting requirement of unknown environment exploration and object picking exploitation given a limited execution horizon. We demonstrate the effectiveness of our framework with software simulation and real-life experiments.**

## I. INTRODUCTION

Recent years have seen great progress in autonomous mobile manipulation applications. Typical examples include bin-picking for construction sites [1], automated public space sanitisation [2] and logistic and warehousing [3]. These applications call for interactive robotic systems that are able to explore the scene while mapping the changing environment and planning their motion and manipulation tasks without colliding with obstacles and with limited on-board resources.

Scene exploration approaches often adopt an information-theoretic strategy that aims to choose the next action to maximise the Information Gain (IG) in an active mapping framework [5]–[7]. For mobile manipulator tasks, active mapping alone is insufficient. Combining mapping and picking in one phase is more effective and efficient than considering them separately. Thus, the task becomes an active and interactive mapping problem, *i.e.* to select and pick the "best" objects with mapping aiding the selection, and where the next movement expands the knowledge of the scene.

This paper presents a computationally efficient environment mapping framework for a mobile manipulator realised on low-budget hardware for practical bin-picking applications, such as is shown in Fig. 1. It aims to address the problems of dynamic scene mapping, exploiting the map's frontier and checking manipulability to select the next best view for efficient mobile base placement and manipulator motion planning. Here, dynamic mapping refers to the accurate capture of environment changes as objects are discovered (scene exploration) and later removed from the scene (object picking) by a mobile manipulator.



(*a*) Initial map and NBV (*b*) Move to NBV and update map

(*c*) Remove object (*d*) Update map and NBV

Fig. 1: The Active and Interactive Mapping cycle. The projection of GPIS mesh on the ground (red line) defines potential exploration segments, green bars indicate segments information utilities. The NBV is a red bar + arrow. Dark green dots are GPIS training points from the removed object.

Our proposed mapping approach is based on Gaussian Probabilistic Implicit Surfaces (GPIS) [12], which encode spatial correlation among input data and offer a probabilistic yet accurate map representation of the world in continuous form. We exploit GPIS to check if a mapped object resides in a robot's workspace. Also, its probabilistic formulation makes it amenable to active mapping based on IG to analytically search for the next-best-view (NBV) and optimal motion.

The contribution of this work is threefold. First, a dynamic GPIS algorithm (Section IV) where each incoming sensor scan forms an instantaneous GP hence defines a probabilistic tolerance layer for valid samples. This detects and discards previous GPIS training points taken from the scene, thus producing a resultant implicit surface accurately capturing environment changes (Fig. 1*(a)* and (*d*)). Secondly, given the probabilistic map representation, we develop an analytical frontier score for each possible NBV candidate (Section V-C) exploiting the GP kernel derivative, this greatly improves fairness and flexibility in NBV selection. Ultimately, the proposed framework is a viable means for a mobile manipulator to interact with an environment (Section V-D): planning base placement and generating arm's trajectory with obstacle avoidance capability, choosing the NBV that maximises object manipulability (Fig. 1(*b*)). Results of the

[1]All authors are with the Center for Autonomous Systems (CAS), University of Technology Sydney, Australia. {Liyang.Liu, Simon.Fryc, Lan.Wu-1, Thanh.Vu, Gavin.Paul-1, Teresa.VidalCalleja}@uts.edu.au

full on-line implementation of our framework are presented in Section VI, demonstrating its performance in simulation and real experiments.

## II. Related Work

**Dynamic mapping**

The choice of mapping representation is critical for object picking tasks with mobile manipulators and needs the ability to update efficiently once objects are removed. The well-known RGB-D mapping system KinectFusion [10] is an example for this purpose. The map is a volumetric type built in the form of Truncated Signed Distance Function (TSDF) data structure. Each voxel in space is time-averaged to smooth out the transient noise. As a side-effect the model handles dynamic scenes. Dramatic topological changes, however, will only appear after a significant delay. Occupancy maps, specifically Octomap [11] are another discrete representation of the 3D world with a probability of occupancy for each voxel in space. A known limitation of the original form is that dynamic environments are not supported. An extension to Octomap was introduced in [1] for a bin-picking application. However, since structural correlations between nearby cells is not addressed, resolution and accuracy in Octomap are often compromised. One can not easily reason about object shape or perform detection [15]. Gaussian Processes Occupancy Maps (GPOM), on the other hand, is a method of combining GP with occupancy mapping that is probabilistic and continuous but has a large computational complexity. Hilbert Maps, first proposed in [26] is a faster and simpler type of occupancy map using kernel approximation methods. Later in [25] it was extended to handle dynamic obstacles and model learning through regression, it finds application in traffic environment modelling.

Despite its advantages, the $O(n^3)$ computational complexity in GPIS [12] has limited its applicability. Lee *et al.* [14] provided an online GPIS implementation that stores data in small clusters for fast parallel processing and incrementally fuses successive scans from various viewpoints. Dynamic environments, however, are not tackled in this method – once an object is mapped, it remains in the GPIS data even if it is physically removed from the workspace. Here, we consider the GP from a fundamental perspective as an immediate probabilistic conditioning tool on current training data [4]. We will show that once a mechanism is devised to detect and filter out training points belonging to the removed object, the map can be instantly updated in the relevant local regions.

Other work in dynamic mapping has been focused on detecting and tracking moving objects in static scenes [8] [9] or eliminating dynamic objects from the environment in order to build accurate static maps [27], which is less applicable to object picking with mobile manipulators.

**Active Exploration**

In active exploration, it is desirable to formulate an IG metric to choose the optimal action. Occupancy maps and Octomaps, with their probabilistic occupancy representation, are often used in mapping and exploration applications [22], [23]. GP, with its continuous uncertainty formulation, has also been used recently for IG-based exploration works, including, entropy reduction [5], conditional entropy reduction [16] and mutual information maximisation [18]. In active shape modelling, [29] proposed an efficient approach using constrained Variational Sparse GP and online kernel learning that preserves reconstruction accuracy.

Entropy gradient [17], is another form of IG metric. The benefit is that it directly draws the robot towards the frontier between the explored and unexplored regions. Authors in [17] devised an approximate gradient formulation for discretised 3D occupancy maps. Later, Jadidi *et al.* [18] generated continuous gradient frontier maps from a 2D GPOM variant, but did not provide an analytical expression due to its non-trivial map definition. They used the gradient map for NBV candidates identification but not in NBV utility calculations. In this paper, we derive the analytical gradient expression from 3D GPIS uncertainty for NBV candidates in 2D mobile manipulator exploration. The simple yet consistent frontier score, together with other IG metrics ensure NBV selection is well-balanced.

**Interactivity**

Interactivity refers here to the robot manipulating objects in the environment. GPIS, with built-in surface normals [15], has been exploited for interactive applications in areas of shape detection, classification and validation. The GPIS-based framework presented in [24] makes use of a robotic arm equipped with tactile sensors to explore and map the environment. Authors in [16] show how to optimise for a reconstruction-aware manipulator trajectory that during a pick-and-place action maximally estimates the object's 3D geometry. Our work, in contrast, generates dense maps and use them for collision-free planning for manipulator trajectory. Further, these prior works assume that every object in the robot's workspace is reachable by the arm. Our framework instead selects the NBV that covers the maximum number of manipulable objects.

## III. System Overview and Problem Statement

The proposed framework aims to provide an efficient pipeline for a mobile manipulator to explore and interact with the environment. As illustrated in Fig. 2, it consists of a GP-based dynamic mapping component that maps the changing world, and an NBV selection module that recommends the best action based on the map information. The robot relies on this framework to plan its arm trajectory and base paths to travel and make changes in the surrounding environment.

Fig. 1 shows a robot's exploration and interactivity cycle. First, it inspects the scene to build the initial map of a multiple objects pile, and computes an NBV that will increase map information. Next, it moves to the recommended NBV and updates the map. Then, it picks up an object and updates the map again to synchronise with the changed world. Based on the updated map, it computes the new NBV and moves towards it, after which the inspection cycle repeats.

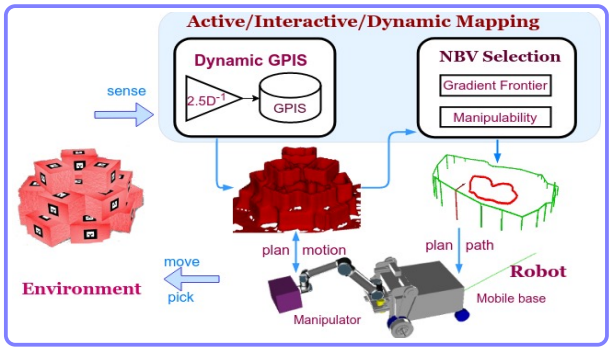



Fig. 2: Active and interactive mapping framework overview.

We now define the notations in this paper: a line segment $\mathbf{s}_i$ describes the robot's pose due to its finite width, $\mathbf{s}_i$ aligns with the base width-wise. The segment set $\mathcal{S} = \{\mathbf{s}_i\}$ denotes candidates in NBV selection. Superscript $\{\cdot\}^{[t]}$ indicates the item is time-dependent, and $\mathbf{s}_i^{[t]}$ is the $i$'th segment at $t$. The sensor pose $\mathbf{T}^{[t]} \in \mathrm{SE}(3)$ is assumed given at all times. $U^{[t]}(\mathbf{s}_i)$ denotes the information utility at time $t$ for the $i$'th segment. The selection goal is to identify the segment $i_*^{[t]}$ with the maximum utility to position the robot at $t$,

$$i_*^{[t]} = \underset{i}{\operatorname{argmax}}\, U^{[t]}(\mathbf{s}_i^{[t]}), \quad i \in \operatorname{supp}(\mathcal{S}^{[t]}) \tag{1}$$

Calculation of a segment's information utility depends on its attributes such as uncertainty, frontier score, and manipulability, all of which revolve around the Dynamic GPIS scene model.

## IV. Dynamic Gaussian Process Implicit Surface

The proposed dynamic mapping representation is based on the online GPIS fusion in [14]. In a similar way, our mapping module consists of two GP phases. A GPIS accumulated from multiple scans, and a frame-level 2.5D$^{-1}$ map as a detector for points removed from the scene. It also exploits independent clusters to store GPIS training points for parallel processing. The main difference with [14] is the dynamic scene handling and introduction of a virtual wall as described in this section.

### A. Continuous distance function as 3D map representation

We now review how GPIS describes scene maps. Let us define a point $\mathbf{x} \in \mathbb{R}^3$ and a function $f_{\text{IS}} : \mathbb{R}^3 \to \mathbb{R}$ such that

$$f_{\text{IS}} = \begin{cases} +d & \text{outside surface} \\ 0 & \text{on surface} \\ -d & \text{inside surface} \end{cases} \tag{2}$$

where $d$ is the point-to-surface distance. The GPIS is defined by the posterior distribution of the value of $f$ at an arbitrary testing point $\mathbf{x}_*$ given by $f(\mathbf{x}_*) \sim \mathcal{N}(\bar{f}_*, \mathbf{P}[f_*])$, where the predictive mean and variance are given by

$$\begin{aligned} f_* &= \mathbf{k}_*^\top (K + K_x)^{-1} \mathbf{y} \\ \mathbf{P}[f_*] &= k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}_*^\top (K + K_x)^{-1} \mathbf{k}_* \end{aligned} \tag{3}$$

$\mathbf{k}_*$, $K$ and $k(\mathbf{x}_*, \mathbf{x}_*)$ represent covariances between $\mathbf{x}_*$ and $n$ training points and $n \times n$ covariance matrix of the training points and the covariance function at $\mathbf{x}_*$, respectively [4]. We use the Matérn 3 class covariance function ($\nu = 3/2$),

$$k_m(d)|_{\nu=3/2} = (1 + \frac{\sqrt{3}d}{l}) \exp(-\frac{\sqrt{3}d}{l}), \quad d = \|\mathbf{x} - \mathbf{x}'\| \tag{4}$$

### B. Instantaneous Scan 2.5D$^{-1}$ Map

For each incoming depth image, a stand-alone elevation map GP is created in the form of bearing angles $\boldsymbol{\theta}$ to inverse depth (IDP) regression. We call it a 2.5D$^{-1}$ map in this paper:

$$f_{\text{IDP}} : \boldsymbol{\theta} \to r^{-1}, \quad \boldsymbol{\theta} = [\theta_u, \theta_v]^T \tag{5}$$

where $r$ is measurement range. IDP is chosen for its i.i.d. Gaussian noise distribution $\eta_{\text{IDP}} \sim \mathcal{N}(0, \sigma_{\text{IDP}}^2)$. Note that every over-limit depth value should be replaced by a large user-defined number. As in [14], the Ornstein-Uhlenbeck (OU) covariance function is used to model IDP observations,

$$k_{\text{OU}}(d) = \frac{1}{2\alpha} \exp(-\alpha d), \quad d = \|\mathbf{x} - \mathbf{x}'\|$$

since the OU kernel is best suited for modelling random walk curves [4] without excessive smoothing.

Given the bearings $\boldsymbol{\theta}_s$, one can infer its inverse depth $r_{\text{IDP}}(\boldsymbol{\theta}_s)$ and uncertainty $\sigma_{\text{IDP}}(\boldsymbol{\theta}_s)$ pair, and obtain coordinates of the corresponding point $\mathbf{x}_s$

$$\begin{aligned} &(r_{\text{IDP}}, \sigma_{\text{IDP}}) = f_{\text{IDP}}(\boldsymbol{\theta}_s), \\ &\mathbf{x}_s = \frac{1}{r_{\text{IDP}}} \mathbf{v}, \quad \mathbf{v} = \frac{\mathbf{v}_h}{\|\mathbf{v}_h\|}, \quad \mathbf{v}_h = \begin{bmatrix} \boldsymbol{\theta}_s & 1 \end{bmatrix}^\top \end{aligned} \tag{6}$$

The uncertainty $\sigma_{\text{IDP}}$ obtained defines a tolerance blanket of allowed range values from the sensor's viewing pose. Any test point in the field of view (FOV), yet falling outside the blanket, is considered an anomaly. This property will be exploited to detect removed objects in Section IV-C.

### C. Data fusion in Dynamic GPs

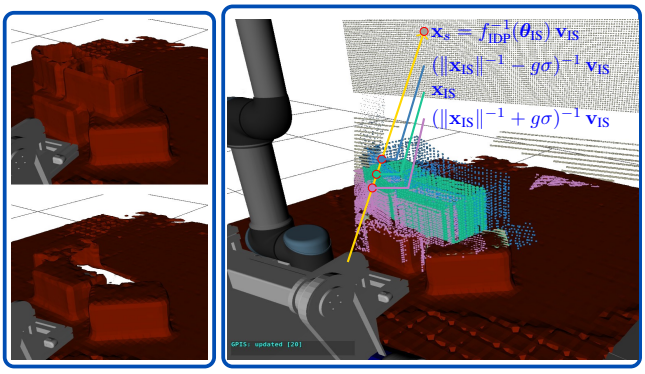
In this stage, the GPIS map is updated with new scan data. We delete samples from the existing GPIS training set that fall outside the tolerance layers defined by the 2.5D$^{-1}$ map. Then fuse or insert new scan points to the GPIS. This results in an immediately clean GPIS data structure that can be conditioned to infer the expanded environment map. The procedure is described in Algorithm 1.

We first identify all GPIS samples that fall into the sensor's FOV at the current pose. For each point in GPIS we transform its coordinates to the local camera's frame $\mathbf{x}_{\text{IS}} = (\mathbf{T}^{[t]})^{-1} \mathbf{x}_{\text{IS}}^{(\text{w})}$. The bearing angles $\boldsymbol{\theta}_{\text{IS}}$ are obtained after homogeneous normalisation. For points inside the FOV, we infer its IDP and uncertainty pair $(r_{\text{IDP}}, \sigma_{\text{IDP}})$ as described in Section IV-B. Then we examine the difference $\triangle_{\text{IS}}$ between the stored inverse depth $r_{\text{IS}}^{-1} = \|\mathbf{x}_{\text{IS}}\|^{-1}$ and the inferred one $r_{\text{IDP}}$:

$$\triangle_{\text{IS}} = r_{\text{IS}}^{-1} - r_{\text{IDP}}(\boldsymbol{\theta}_{\text{IS}}), \tag{7}$$

to apply three possible treatments: delete, fuse or ignore.

**Delete:** For regions containing removed objects, the new scan reveals what is lying behind the old object along its light




(*a*) Map before and after object remove

(*b*) Delete point $\mathbf{x}_{\text{IS}}$ if range significantly different from its 2.5D$^{-1}$ correspondence: $\|\mathbf{x}_{\text{IS}}\|^{-1} - f_{\text{IDP}}(\frac{\mathbf{x}_{\text{IS}}}{\|\mathbf{x}_{\text{IS}}\|}) \gg \sigma_{\text{IDP}}$.

Fig. 3: Map update in Dynamic GP. Using ray-casting a reverse tolerance layer is formed around GPIS points: $+\sigma_{\text{IDP}}$ (blue) and $-\sigma_{\text{IDP}}$ (violet). Removed object points (green) are detected and deleted from GPIS, forming a new map. The 2.5D$^{-1}$ map with the virtual wall at "infinity" is shown in dark yellow. An example FOV ray is shown in yellow

**Algorithm 1:** Dynamic GPIS Update

$f_{\text{IDP}} \leftarrow \text{Regress2.5D}^{-1}(\text{Scan}^{[t]})$;
**for** $\mathbf{x}_{\text{IS}}^{(\text{w})} \in \text{GPIS}^{[t]}$ **do**
  $\mathbf{x}_{\text{IS}} = (\mathbf{T}^{[t]})^{-1}\,\mathbf{x}_{\text{IS}}^{(\text{w})}$;
  $(r_{\text{IDP}}, \sigma_{\text{IDP}}) \leftarrow f_{\text{IDP}}(\boldsymbol{\theta}_{\text{IS}})$ ; // Eq. (5)
  $\triangle_{\text{IS}} \leftarrow \text{Compare}(f_{\text{IDP}}, \|\mathbf{x}_{\text{IS}}\|^{-1})$ ; // Eq. (7)
  **if** $\triangle_{\text{IS}} \geq g\sigma_{\text{IDP}}$ **then**
    /* point removed from scene */
    $\mathbf{x}_{\text{IS}}^{(\text{w})} \leftarrow \varnothing$ ;
  **else**
    **if** $-g\sigma_{\text{IDP}} \leq \triangle_{\text{IS}} \leq g\sigma_{\text{IDP}}$ **then**
      /* old and new fusable */
      $\mathbf{x}_s \leftarrow \text{Invert}(r_{\text{IDP}}, \boldsymbol{\theta}_{\text{IS}})$ ; // Eq. (6)
      $\mathbf{x}_{\text{IS}}^{(\text{w})} \leftarrow \mathbf{T}^{[t]}\,\text{Fuse}(\mathbf{x}_{\text{IS}}, \mathbf{x}_s)$ ; // Ref. [14]
    **end**
  **end**
**end**
$\text{GPIS}^{[t+1]} \leftarrow \text{GPIS}^{[t]} \bigcup \text{Scan}^{[t]}$;

ray. The range measured would be either larger or outside the sensor range. We delete this point using the anomaly detection criterion, $\triangle_{\text{IS}} \geq g\,\sigma_{\text{IDP}}$, where $g$ is a constant scale factor. However, this simple check fails if the background is empty since it destroys the 2.5D$^{-1}$ map model. Here, using a method similar to [1], we artificially replace every over-limit measurement in the scanned depth with a very large number, effectively creating a virtual wall at "infinity", see Fig. 3 for illustration. The virtual wall is processed together with the rest of the scan data to form the 2.5D$^{-1}$ map, allowing valid inferences for every bearing angle.

**Fuse:** For $\triangle_{\text{IS}}$ within a small range, the old GPIS point $\mathbf{x}_{\text{IS}}$ will be fused with its corresponding point on the 2.5D$^{-1}$ surface as described in [14].

**Ignore:** Points outside the above categories are occluded scene points and should be left unchanged.

With the old GPIS training points cleaned up, the new scan samples are now added for map expansion. The modified local GPIS clusters from both ends are processed further (covariance inversion) for fast future inference. For a full treatment of online GPIS formulation please refer to [14].

## V. NBV Selection

To select an optimal next pose, we first identify a set of candidate poses, then compute its utility function, incorporating various GPIS based metrics including gradient frontier and manipulability.

### A. Candidate Poses

Since the robot needs to explore objects in a pile, the next position to place the robot should be around the circumference of the partially explored pile, as shown in Fig. 4. We first query the GPIS to obtain a probabilistic implicit surface representing the scene, then use the marching cubes algorithm [28] to form a dense map. The map can be assumed to have a roughly conical shape which is generically the case when stacking objects to maintain stability. We project the surface points onto the ground to form a 2D occupancy map. Using image processing methods, we obtain the contour of the occupancy, (see Fig. 4(*a*)), which is composed of a set of piece-wise linear segments $\{\mathbf{s}_i\}$ of the map base. These segments are the candidates for positioning the robot in the next step. Half of $\{\mathbf{s}_i\}$ are from the unseen surface area without full exploration, hence have high uncertainty, referred to as "imaginary". The remaining half are genuine base outline segments and have low uncertainty. Using this uncertainty we classify each segment $\mathbf{s}_i$ as real or imaginary.

### B. Utility Formulation

Many factors (or attributes) from $\mathbf{s}_i$ can affect the utility. We use $\{\cdot\}[\mathbf{s}_i^{[t]}]$ to denote attribute $\{\cdot\}$ of segment $i$. For the sake of simplicity, from now on we omit the time superscript in segment attributes. These attributes are:

- **Uncertainty** $\sigma^2[\mathbf{s}_i]$: encourages information collection for noisy regions. Only the real segments are considered here. The imaginary ones are addressed below.
- **Frontier** $f_{\nabla\sigma^2}[\mathbf{s}_i]$: for unexplored regions, frontier gives preference to the boundary segments over other imaginary segments.
- **Arm manipulability** $m[\mathbf{s}_i]$: This factor gives a quantitative measure of volume in GPIS that is reachable by the manipulator.
- **Interact order** $h[\mathbf{s}_i]$: gives a preference for picking order and can be in the form of segment height, readily available from the GPIS. This order factor is useful since picking in an order from high to low positions causes minimum disruption to the object pile. Further, this encourages the pile to maintain a convex-shaped outline, convenient for base navigation.
- **Travel distance** $d[\mathbf{s}_i]$: preference is given to visit close-by scanned segments. It is computed as the Dijkstra

minimum distance from current location to candidate.

- **Avoid repeated failure** $p(\,\mathbf{s}_i, t\,)$: avoids locations where previous pick/detect attempts failed within a time duration, and is a function of time.

Considering these factors, we present the following utility formulation to select the optimal candidate segment $i_*$:

$$\begin{aligned} i_*^{[t]} &= \underset{i}{\text{argmax}}\, U^{[t]}(\,\mathbf{s}_i\,), \quad i \in \text{supp}(\,\mathcal{S}^{[t]}\,), \\ U^{[t]}(\,\mathbf{s}_i\,) &= (1 - p(\,t\,)[\mathbf{s}_i])\,\mathcal{I}(\,\mathbf{s}_i\,), \\ \mathcal{I}(\,\mathbf{s}_i\,) &= \beta_1\,\mathcal{L}_1(\,m[\mathbf{s}_i]\,) + \beta_2\,\mathcal{L}_2(\,h[\mathbf{s}_i]\,) \\ &\quad + \beta_3\,\mathcal{L}_3(\,-d[\mathbf{s}_i]\,) + \beta_4\,\mathcal{L}_4(\,\sigma^2[\mathbf{s}_i]\,) \\ &\quad + \beta_5\,\mathcal{L}_5(\,f_{\nabla\sigma^2}[\mathbf{s}_i]\,), \\ 1 &= \sum_k \beta_k, \quad \beta_k \geq 0, \\ p(\,t\,)[\mathbf{s}_i] &= \gamma^{\,t - t_f[\mathbf{s}_i]}, \quad \gamma < 1 \\ \mathcal{S}^{[t+1]} &\leftarrow \text{action}(\; i_*^{[t]} \;) \end{aligned} \tag{8}$$

where

$$\begin{aligned} \mathbf{s}_i &:= (\,m,\, \sigma^2,\, f_{\nabla\sigma^2},\, h,\, d,\, p(t)\,)[\mathbf{s}_i], \\ \mathcal{I}(\cdot) &: \text{information gain} \\ \forall\, k &= 1, 2, 3, 4, 5 \\ \beta_k &: \text{ user defined weightings} \\ \mathcal{L}_k(\,x\,) &: \text{ logistic function, maps to probability} \\ &= \frac{l_k}{1 + \exp(\,-a_k x + b_k\,)}, \\ (l_k, a_k, b_k) &: \text{empirically obtained parameters} \end{aligned} \tag{9}$$

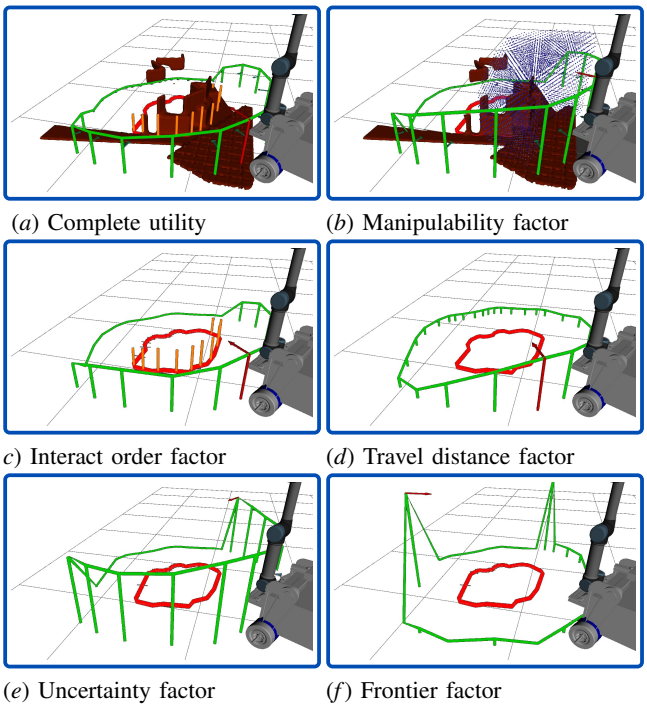
Note that $p(\mathbf{s}_i, t)$ takes the form of *discounted* future penalty. It relies on the duration between the last failure time $t_f$ and the current time. Initially we set $t_f$ to $-\infty$ for zero penalty and update it once a failure occurs. This activates the penalty and deactivates the segment for a small time frame.
During runtime, all imaginary segments have their manipulability, height, and uncertainty set to zero, resulting in zero utility. This prevents the robot from landing on unknown regions. The frontier segments with valid $m$, $h$ and $\sigma^2$ scores, compete with other real segments for NBV selection. Once the frontier is explored, its surrounding imaginary regions become "real" with the frontier shifted. The new segments are added to the next round of NBV selection.

### C. Gaussian Process Frontier

We now propose our GP based frontier metric which gives higher precedence for boundary segments over unexplored regions. The uncertainty gradient for an arbitrary point $\mathbf{x}$ can be defined as

$$\|\nabla\sigma^2(\mathbf{x})\|^2.$$

The robot base motion is confined to the 2D ground plane, during manipulation the base is aligned width-wise with the pile segment. Due to the segment's finite length, it is necessary to integrate the gradients along the segment direction to obtain the overall uncertainty variation.
Let $\mathbf{l}_i = [l_x, l_y, 0]^\top$ denote the direction of segment $\mathbf{s}_i$,



(*a*) Complete utility (*b*) Manipulability factor
(*c*) Interact order factor (*d*) Travel distance factor
(*e*) Uncertainty factor (*f*) Frontier factor

Fig. 4: An illustration of full formulation and single factor utilities. First, identify pile segments (red contour) and compute the utilities (green bars). Max utility gives NBV (red bar+arrow). (*b*) shows samples (blue dots) from the manipulability annulus. (*c*) shows height (orange bar). Grey bars on the ground are uncertainty for imaginary segments.

transversal to the robot heading. Let $\mathcal{P}_i$ denote the set of 3D points in $\mathbf{s}_i$. Denote $\delta\alpha$ as the infinitesimal segment length along $\mathbf{l}_i$. Then for a point $\mathbf{x}_j^i \in \mathcal{P}_i$, a small perturbation results in a neighbour point $\mathbf{x}_j^i + \delta\alpha\mathbf{l}_i$. We define the frontier metric as the Sum of Directional Squared Difference (SDSD) in uncertainty for all points in $\mathcal{P}_i$ along $\mathbf{l}_i$:

$$\begin{aligned} f_{\nabla\sigma^2} &= \sum_{\mathbf{x}_j^i \in \mathcal{P}_i} \left\| \frac{\partial(\,\sigma^2(\mathbf{x}_j^i + \delta\alpha\mathbf{l}_i) - \sigma^2(\mathbf{x}_j^i)\,)}{\partial\delta\alpha} \right\|^2 \\ &= \sum_{\mathbf{x}_j^i \in \mathcal{P}_i} \left\| \frac{\partial(\,\sigma^2(\mathbf{x}_j^i) + \delta\alpha\nabla\sigma^2(\mathbf{x}_j^i)^\top\,\mathbf{l}_i - \sigma^2(\mathbf{x}_j^i)\,)}{\partial\delta\alpha} \right\|^2 \\ &= \sum_{\mathbf{x}_j^i \in \mathcal{P}_i} \left\| \nabla\sigma^2(\mathbf{x}_j^i)^\top\,\mathbf{l}_i \right\|^2 \end{aligned} \tag{10}$$

For gradient at point $\mathbf{x}_j^i$, we use GPIS definition (3) and Matérn kernel (4) to compute as follows:

$$\begin{aligned} \nabla\sigma^2(\mathbf{x}_j^i)^\top &= 2\,\mathbf{k}_*^\top\,(K + K_x)^{-1}\,\nabla\mathbf{k}_*(\mathbf{x}_j^i)^\top \\ \nabla\mathbf{k}_*(\mathbf{x}_j^i)^\top &= \left[\,\ldots,\quad \nabla k_m(\mathbf{x}', \mathbf{x}_j^i)^\top,\quad \ldots\right]^\top \\ \nabla k_m(\mathbf{x}', \mathbf{x}_j^i)^\top &= \frac{\partial k_m(d)}{\partial d}\,\frac{\partial d}{\partial \mathbf{x}_j^i} \end{aligned} \tag{11}$$

where: $d = \left\| \mathbf{x}_j^i - \mathbf{x}' \right\|$, $\dfrac{\partial k_m(d)}{\partial d} = -\dfrac{3d}{l^2}\exp(-\dfrac{\sqrt{3}d}{l})$.

This GP-based definition is continuous and mathematically rigorous. It contrasts to the discretised frontier metric in [17].

Its simple formulation is more tractable and deployable than [18], which is only for 2D maps and relies on an auto-jacobian method from optimisation libraries, due to the hybrid GPOM world model used.

### D. Manipulability

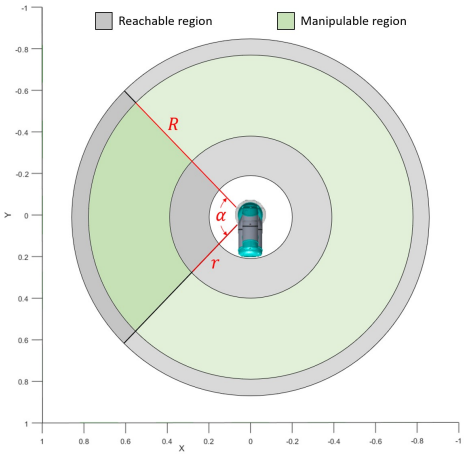



Fig. 5: A top-view of the annulus sector used in the pose manipulability filter.

This section presents a metric to determine whether the manipulator is capable of reaching and picking the objects in the environment. This metric filters out object poses that minimise the manipulability index [19] of the arm during picking. The filter uses the geometric volume of an annulus (donut-like) sector as a heuristic to determine which poses are viable. The frame of this region is fixed to the manipulator base frame and all poses outside this region are rejected by the filter. For this process, we use the determinant of the Jacobian matrix $\mathbf{J}$ to calculate the robot's measure of manipulability $m$ as follows,

$$m = \sqrt{\det(\mathbf{J}\mathbf{J}^T(\theta))} \in \mathbb{R} \tag{12}$$

where a robot's Jacobian $\mathbf{J}$ relates an end-effector's Cartesian velocity with the joint velocities $\dot{\mathbf{q}}$. The annulus sector used in the pose manipulability filter is derived from the manipulator's kinematics by sampling valid end-effector discrete poses. For each configuration $\mathbf{q}$ in the configuration space, we calculate the end-effector pose $\xi_E$ in task space $\mathcal{T}$ using forward kinematics and the manipulability index $m$ where:

$$\xi_E = (R, \mathbf{p}) \in \mathcal{T}, \quad \mathcal{T} \subset \mathrm{SE}(3), \tag{13}$$

with the position $\mathbf{p} \in \mathbb{R}^3$ and the rotation $R \in \mathrm{SO}(3)$.

We select only the configurations with manipulability index $m \geq m_{thres}$. The remaining points are used to determine the minimum and maximum bounds of the manipulable Cartesian workspace $\mathcal{T}_m \subset \mathcal{T}$. This allows us to differentiate between the reachable region of the manipulator workspace and the manipulable region of the workspace as shown in Fig. 5, where the manipulable region in front of the robot is defined with a radius interval $[r, R]$, height interval $[h, H]$, and an angle $\alpha$. The values of $[h, H]$ and $\alpha$ are chosen by plotting all valid configurations.

With a robot located at the $i$'th pile segment, we define the segment's manipulability as the overlapping region between an arm's annulus sector and the mapped pile. Specifically, for each sample point in the annulus sector, we query the GPIS for its occupancy probability [13] using its inferred signed distance $\mu$, variance $\sigma^2$ and normal $\mathbf{n}$. The sum of weighted occupancy for all samples defines the manipulability score:

$$\begin{aligned} m[\mathbf{s}_i] &= \sum_{\mathbf{p}_j \in \mathcal{T}_m} w_j \, p(o=1|\mathbf{p}_j) = \sum w_j \Phi\left(\frac{\alpha\mu_j + \beta}{\sqrt{1+\alpha^2\sigma_i^2}}\right); \\ \begin{bmatrix}\mu_j \\ \sigma_j \\ \mathbf{n}_j\end{bmatrix} &= f_{\mathrm{IS}}(\mathbf{p}_j^{(\mathrm{w})}), \quad w_j = \frac{\mathbf{n}_j \cdot \mathbf{p}_j^{(\mathrm{w})}}{\|\mathbf{p}_j^{(\mathrm{w})}\|}, \quad \mathbf{p}_j^{(\mathrm{w})} = \mathbf{T}[\mathbf{s}_i]\,\mathbf{p}_j \end{aligned} \tag{14}$$

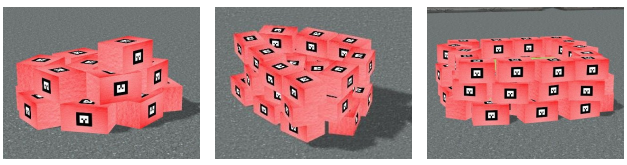
The more objects reachable by the robot in alignment with its orientation at the segment, the higher its $m$ score. Fig. 4(*b*) shows an example of the pose with highest $m$ score.

## VI. Evaluation

We evaluated the performance of our framework with extensive simulated and real-life experiments. We built our mobile manipulator using Neobotix MP700 [20] as the base and URe5 [21] for the arm with a magnetic contact end-effector. An RGB-D camera is mounted on the mobile base. Our active mapping framework is developed according to the description in Section III. It is written in C++/ROS and runs on a 6-core laptop. Our test environment consists of piles of bricks on flat terrain. All piles are roughly 4m$^2$ in size with ∼50 segments considered in the optimisation, which is trivially performed by finding the maximum amongst all candidate utility scores. The bricks are labelled with AR-tags for easy pose detection with metal plates attached for magnetic grasping. This scenario resembles the necessary exploration and an object picking component in automatic construction tasks. A video accompanying our results can be found in https://tinyurl.com/y5qwn864.

### A. Simulation

A simulated Gazebo environment was created to contain brick piles as shown in Fig. 6. Gazebo models of the base and arm were also designed with accurate mechanical properties to mimic the real-life system. An ablation study and benchmark tests are performed for the simulated environment.



(*a*) simu case 1 (*b*) simu case 2 (*c*) simu case 3

Fig. 6: Three Gazebo simulation scenarios.

*1) Ablation Study:* We analysed the importance of each factor in our NBV selection scheme, by removing it from the framework and observing the performance results as given in Table I. We run each variant in a fixed time interval and collect the average number of picked bricks, percentage of covered map, number of object falls and collisions between robot and environment. The "Pick Order" row is important in maintaining a balanced pile shape, failing to do that the test has a high collapse count as seen in the "Falls" column – a scenario to be avoided for warehousing or construction applications. The completion rate in "Travel Distance" and "Frontier" rows are related to collisions. The robot should never enter an unknown region before mapping it. Further, without the frontier factor, map coverage is affected; hence, achieving 100% can take a protracted time. Without the "Failure Penalty" factor, the robot can get stuck indefinitely on a seemingly good spot that does not host pick-able objects. Considering all results, the complete utility function (row 1) case performs the best in each test category.

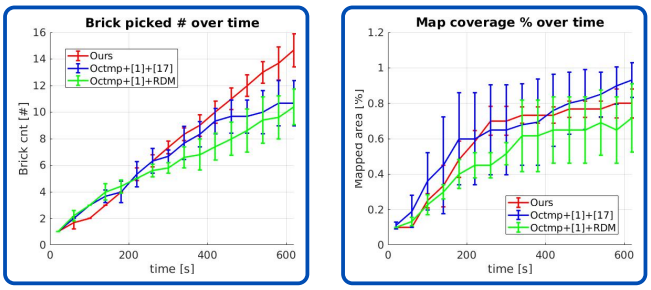
TABLE I: Ablation analysis on Utility factors.

| Part removed | Output | | | |
|---|---|---|---|---|
| | Bricks # | Map % | Falls # | Collision ? |
| NONE | **17** | **100** | **1** | **N** |
| Manipulability | 14.5 | 100 | 2 | N |
| Pick Order | 13 | 100 | 6 | N |
| Travel Distance | 11 | 80 | 3 | Y |
| Uncertainty | 15 | 100 | 1 | N |
| Frontier | 16 | 90 | 4 | Y |
| Failure penalty | 7 | 60 | 1 | N |

*2) Benchmark Test:* We performed benchmark testing by comparing task throughput and map coverage between our system and two other systems for three test scenarios, five times each (Fig. 6). The other systems are: *(1)* Octomap + [1] (dynamic) + [17] (discrete gradient frontier), and *(2)* Octomap + [1] (dynamic) + random (RDM) strategy. The results are shown in Table II, showing the number of bricks picked (task) and map coverage. From the results, our framework outperforms *(1)* and *(2)* in mission completion rate. We observed in *(1)* and *(2)* that the robot frequently went to locations that do not host pick-able objects. This can be explained by the fact that our framework uses a GPIS-based manipulability factor that chooses more promising locations for picking. *(1)* has the best map coverage but can occasionally lead the robot into the obstacle zone, this can be explained by its approximate frontier calculation. (*2*) has the lowest map coverage as it tends to stay in already explored regions. We also illustrate the execution progress of the three systems for test case 2 in Fig. 7. Our approach has the best performance for the picking object task and equivalent performance for map coverage, compared to Octomap +[1]+[17], but with smaller variations over the different runs.

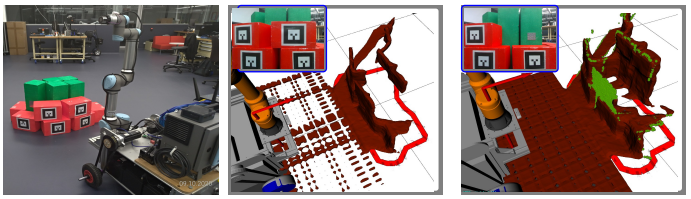
### B. Real-life Experiment

In the real-life experiment, we tested the accuracy of the dynamic GPIS mapping component and the effectiveness of NBV selection in our framework.

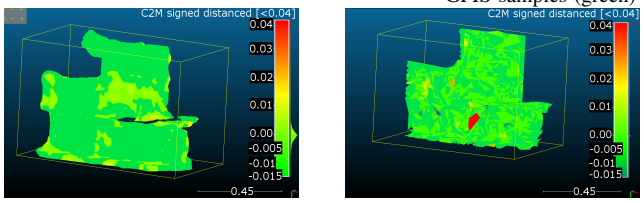



(*a*) Objects picked (*b*) Map coverage %

Fig. 7: Comparison of task and map coverage for simu case 2.



(*a*) Scene set up (*b*) Initial GPIS (*c*) Remove brick, delete GPIS samples (green)




(*d*) Dynamic GPIS map vs [14], (note the smooth surface), obtain distance errors: $\mu_{dist} = 0.005$ cm, $\sigma_{dist} = 0.012$

(*e*) Apply [1], obtain before and after Octomaps, (note the artifacts), obtain distance errors: $\mu_{dist} = 0.033$ cm, $\sigma_{dist} = 0.015$

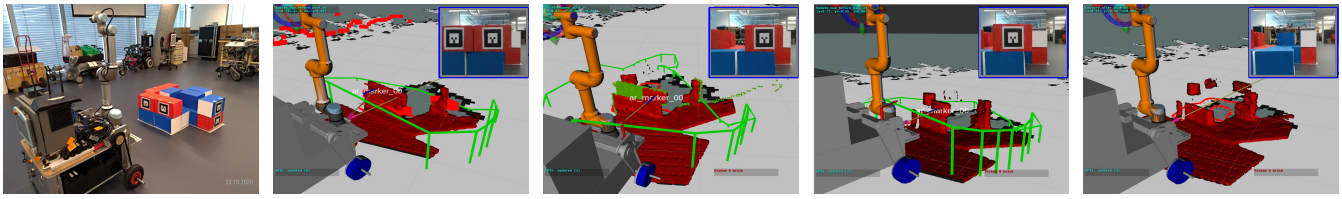
Fig. 8: Real-life experiment: comparing map accuracy between ours, [14] and [1].

*1) Dynamic GPIS accuracy:* We set up a real-life scene Fig. 8(*a*) to evaluate the accuracy of dynamic GPIS by comparing with the methods of [14] and [1]. We first generated a map using our Dynamic GPIS with a 2-step process: initialise using the original scene (*b*), then update after removing a brick (*c*). For reference, we generated a map by feeding the depth image in step 2 directly to [14]. We compared the signed distance values in the two maps using CloudCompare (*d*), and the error was insignificant (*e*). Further, we applied the same procedure for the Octomap variant [1] and our results are shown to be superior (*e*).

*2) NBV test:* In real-life test scene 2 (Fig. 9(*a*)), we evaluated the effectiveness of our NBV selection. We set the scene to have two rows of bricks with the top two lying side by side. Initially, the robot was set to face the top two bricks (*b*). Then the robot detected and picked the top left brick, the map was updated and the NBV shifted towards the high brick on the right (*c*). The robot then moved to the NBV

TABLE II: Benchmark Test

| Simu case | Ours | | *(1)* Octmp+ [1]+ [17] | | *(2)* Octmp+ [1]+RDM | |
|---|---|---|---|---|---|---|
| | *Bricks%* | *Map%* | *Bricks%* | *Map%* | *Bricks%* | *Map%* |
| #1 | **95**±5.0 | **100**±10 | 63±17 | **100**±5.0 | 45±12 | 90±10 |
| #2 | **100**±6.0 | 82±8.5 | 72±8.0 | **85**±9.0 | 61±12 | 71±19 |
| #3 | **100**±3.0 | 61±3.5 | 66±13 | **70**±16 | 57±6.8 | 57±12 |



(*a*) test scene 2 (*b*) first scan (*c*) picked one brick (*d*) moved to NBV (*e*) picked another brick

Fig. 9: Real-life experiment, active and interactive mapping cycle.

position, (*d*). Finally, it picks up the right brick. This shows our NBV strategy behaved in the desired order.

## VII. Conclusion

We presented an interactive and active mapping framework for a mobile manipulator platform based on dynamic GPIS. Since the framework is probabilistic, it is able to perform immediate mapping updates for a dynamically changing environment. Using the probabilistic map, our NBV selection scheme has been shown to balance the needs of information gain in visited regions, frontier driven map expansion, as well as object manipulability. Most importantly, the dense map generated enables a robot to safely move around in, and apply changes to, the environment. Both simulation and real-life experiments show our system can efficiently explore and interact with a large pile of objects in an environment.

## References


[1] S. Fryc, L. Liu, and T. Vidal-Calleja, "Efficient pipeline for mobile brick picking," in *Australasian Conference on Robotics and Automation - (ACRA)*, 2019.

[2] IEEE, "Robotic Arm Wields UV Light Wand To Disinfect Public Spaces," 2020. [Online]. Available: https://spectrum.ieee.org/news-from-around-ieee/the-institute/ieee-member-news/usc-researchers-robotic-arm-disinfect-coronavirus

[3] G. Bartels and M. Beetz, "Perception-Guided Mobile Manipulation Robots for Automation of Warehouse Logistics," *KI - Künstliche Intelligenz*, vol. 33, pp.189–192, 2019.

[4] C. E. Rasmussen and C. K. I. Williams, "Gaussian Processes for Machine Learning". MIT Press, 2006.

[5] M. Popović, T. Vidal-Calleja, G. Hitz, I. Sa, R. Siegwart, and J. Nieto, "Multiresolution mapping and informative path planning for UAV-based terrain monitoring," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1382–1388.

[6] M. Lauri, N. Atanasov, G. J. Pappas, and R. Ritala, "Active object recognition via monte carlo tree search," in *International conference on robotics and automation (ICRA) Workshop* , 2015.

[7] F. Sukkar, G. Best, C. Yoo, and R. Fitch, "Multi-robot region-of-interest reconstruction with dec-mcts," in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 9101–9107.

[8] C. Jiang, D. Paudel, Y. Fougerolle, D. Fofi, and C. Demonceaux, "Static-map and dynamic object reconstruction in outdoor scenes using 3-d motion segmentation," *IEEE Robotics and Automation Letters*, vol. 1, pp. 1–1, 01 2016.

[9] B. Bescos, J. M. Facil, J. Civera, and J. Neira, "Dynaslam: Tracking, mapping, and inpainting in dynamic scenes," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, p. 4076–4083, Oct 2018.

[10] R. A. Newcombe, S. Izadi, O. Hilliges, D. Kim, A. J. Davison, P. Kohli, J. Shotton, S. Hodges, and A. Fitzgibbon, "Kinectfusion: Real-time dense surface mapping and tracking," in *IEEE ISMAR*. IEEE, October 2011.

[11] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Autonomous Robots*, 2013, software available at http://octomap.github.com.

[12] O. Williams and A. Fitzgibbon, "Gaussian process implicit surfaces," 2007.

[13] S. Kim, J. Kim. "Continuous Occupancy Maps Using Overlapping Local Gaussian Processes," *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*.

[14] B. Lee, C. Zhang, Z. Huang, and D. D. Lee, "Online continuous mapping using gaussian process implicit surfaces," in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 6884–6890.

[15] S. Dragiev, M. Toussaint, and M. Gienger, "Gaussian process implicit surfaces for shape estimation and grasping," in *2011 IEEE International Conference on Robotics and Automation*, pp. 2845–2850.

[16] K. Huang and T. Hermans, "Building 3d object models during manipulation by reconstruction-aware trajectory optimization," *ArXiv*, vol. abs/1905.03907, 2019.

[17] R. Rocha, J. Dias and A. Carvalho, "Cooperative Multi-Robot Systems A study of Vision-based 3-D Mapping using Information Theory," *2005 IEEE International Conference on Robotics and Automation*, pp. 384-389.

[18] M. Jadidi, J. Miro and G. Dissanayake, "Gaussian processes autonomous mapping and exploration for range-sensing mobile robots," *Autonomous Robots*, no. 42, pp. 273–290, 2018.

[19] T. Yoshikawa, "Manipulability of robotic mechanisms," *The International Journal of Robotics Research*, vol. 4, no. 2, pp. 3–9, 1985.

[20] N. GmbH, "Mobile robot mp-700," 2020. [Online]. Available: https://www.neobotix-robots.com/products/mobile-robots/mobile-robot-mp-700

[21] "Universal Robot UR5e, a flexible and lightweight robotic arm," 2020. [Online]. Available: https://www.universal-robots.com/products/ur5-robot

[22] Z. Zhang, T. Henderson, S. Karaman and V. Sze, "FSMI: Fast computation of Shannon mutual information for information-theoretic mapping," *The International Journal of Robotics Research*, vol.39, no. 9, pp. 1155–1177, 2020.

[23] F. Bourgault, A. Makarenko, S. Williams, B. Grocholsky and H. Durrant-Whyte, "Information based adaptive robotic exploration," *2002 International Conference on Intelligent Robots and Systems*, pp. 540-545 vol. 1.

[24] S. Caccamo, Y. Bekiroglu, C. H. Ek and D. Kragic, "Active exploration using Gaussian Random Fields and Gaussian Process Implicit Surfaces," *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

[25] R. Senanayake, and F. Ramos. "Bayesian Hilbert Maps for Dynamic Continuous Occupancy Mapping," *Proceedings of the 1st Annual Conference on Robot Learning*, vol. 78, pp. 458-471.

[26] F. Ramos, L. Ott. "Hilbert maps: Scalable continuous occupancy mapping with stochastic gradient descent." *The International Journal of Robotics Research*, 35(14), 2016.

[27] T. Hahnel, "Map building with mobile robots in dynamic environments," *2003 IEEE International Conference on Robotics and Automation*, vol. 2, pp. 1557–1563.

[28] W. Lorensen and H. Cline. "Marching cubes: A high resolution 3D surface construction algorithm," 1987 *SIGGRAPH Comput. Graph*, vol. 21, pp. 163–169.

[29] G. Gandler, C. Ek, M. Björkman, R. Stolkin and Y. Bekiroglu, "Object shape estimation and modeling, based on sparse Gaussian process implicit surfaces, combining visual data and tactile exploration", *Robotics and Autonomous Systems*, vol. 126, 2020.