CHALMERS
UNIVERSITY OF TECHNOLOGY

Detecting Changes on the ISS Autonomously with 3D Point Clouds

An Unsupervised Learning Approach Using GMM Clustering

Master's Thesis in Complex Adaptive Systems

JAMIE SANTOS

DEPARTMENT OF MECHANICS AND MARITIME SCIENCES

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2023
www.chalmers.se

MASTER'S THESIS IN COMPLEX ADAPTIVE SYSTEMS

Detecting Changes on the ISS Autonomously with 3D Point Clouds

JAMIE C. SANTOS

Department of Mechanics and Maritime Sciences
Division of Vehicle Engineering and Autonomous Systems
CHALMERS UNIVERSITY OF TECHNOLOGY

Göteborg, Sweden 2023

Detecting Changes on the ISS Autonomously with 3D Point Clouds
JAMIE C. SANTOS

Department of Mechanics and Maritime Sciences
Division of Vehicle Engineering and Autonomous Systems
Chalmers University of Technology
SE-412 96 Göteborg
Sweden
Telephone: +46 (0)31-772 1000

Cover:
A view of Astrobee within the Granite Lab at NASA Ames Research Center, where data was collected for testing

Chalmers Reproservice
Göteborg, Sweden 2023

Detecting Changes on the ISS Autonomously with 3D Point Clouds
Master's thesis in Complex Adaptive Systems
JAMIE C. SANTOS
Department of Mechanics and Maritime Sciences
Division of Vehicle Engineering and Autonomous Systems
Chalmers University of Technology

Abstract

New space habitats, such as the Lunar Gateway and subsequent stations on Mars, will be increasingly difficult to continuously staff with astronauts, necessitating robotic support in the absence of humans. In this work, an unsupervised change detection algorithm requiring only 3D depth data as input is proposed in order to enable autonomous robotic caretaking. Scene change detection is necessary for robotic caretakers to accomplish many routine tasks, such as map updating and surveillance of the habitat. Upon the International Space Station (ISS), Astrobee is one such robot used for development and demonstration of these technologies under the Integrated System for Adaptive Autonomous Caretaking (ISAAC) project. Current scene analysis developed within the ISAAC project uses semantic localization to detect manually labeled objects within the map. However, this is not a sustainable approach for generalized change detection, as thousands of images must be captured and labeled for accurate results. In contrast, the proposed algorithm uses an unsupervised GMM clustering algorithm to compare "before" and "after" point clouds of the scene, and is therefore capable of detecting changes in the scene without being restricted to manually labeled objects. Experiments with data collected at NASA Ames' Granite Lab, a mock-up of the ISS, successfully demonstrate the detection of one or more object appearances or disappearances in the scene with an initial average F1 score of 74% for volumetric reconstructed maps of the scene with two and three changed objects, and 70% for the comparison of single frames from the depth camera with one object. With a number of optimizations that can be made to improve the accuracy, the source code is released to the public to promote further research and development.

Keywords: robotics, Gaussian mixture model, earth mover's distance, change detection, unsupervised learning, expectation maximization, point cloud, ISS

Acknowledgements

This thesis was completed as an internship within the Intelligent Robotics Group at NASA Ames Research Center in Mountain View, California. Thank you to Dr. Brian Coltin, Dr. Trey Smith, and Marina Moreira Gouveia for their mentorship and guidance throughout this project. I would also like to express gratitude to Dr. Paulo V.K. Bourges and Holly Dinkel for letting me bounce ideas off of them and for providing moral support. Thank you to Professor Mattias Wahde of Chalmers University for project oversight and feedback. And last but not least, thank you to my mom, dad, brother, Zaijen, friends, and my partner, Chandler, for their love and support as I pursue my dreams.

Jamie Santos, Göteborg, June 2023

Nomenclature

D	=	the dimensions of the parameter space of a Gaussian
$\mathbf{D}_{K_{t_0}}$, $\mathbf{D}_{K_t}$	=	the matrix representing the metric distance between all means in $\mathbf{\Theta}^{t_0}$ and $\mathbf{\Theta}^t$
K	=	the total number of Gaussian clusters in a Gaussian Mixture Model (GMM)
$\mathbf{\Pi}$	=	the Gaussian mixture model representing the areas within the map of where changes occurred
$\mathbf{S}^{t_0}, \mathbf{S}^t$	=	the set of 3D points representing a scene at time t
t_0, t	=	the two times at which the state of the scene is captured
$\mathbf{\Theta}^{t_0}$, $\mathbf{\Theta}^t$	=	the Gaussian mixture model representing the map at t

Contents

1 Introduction

Anomaly detection is a critical task in the field of robotics. Applications include map updating [Der+21], surveillance [Wel+17], and manufacturing [CP16]. These applications are also embedded within the context of robotic caretaking in space stations and habitats. Changes detected in the station can be used to highlight locations to remap for improved localization. Surveillance is also crucial for maintaining a safe and stable environment for the health of inhabitants and space stations alike. The capacity to search for anomalies will enable systems to detect critical safety issues such as blocked vents or loose cargo. Furthermore, in the absence of humans, robots may be required to assemble objects autonomously, in which case anomaly detection may be used for tasks such as identifying damaged components. This work focuses on the needs of the former two cases, in which an algorithm that autonomously detects generalized changes within a scene is desired.

One such robotic caretaking platform is currently already underway at NASA Ames Research Center. The Integrated System for Autonomous and Adaptive Caretaking (ISAAC) software project will serve as a foundation for autonomously maintaining space stations when human presence cannot be sustained, and to also support astronauts while they are inhabiting spacecraft. Currently, ISAAC runs on the Astrobee free-flying robot aboard the International Space Station (ISS). Previous anomaly detection work within ISAAC has been semantic- and image-based. The key drawback with semantic-based methods is that changes can only be detected on items that have been through an extensive labeling process, currently entirely by hand. While this strategy works well for key targets such as hazard labels or handrails, the use of semantic processing will rely on up-to-date labeling and will be limited to the number of classes added manually. Furthermore, a major limitation of image-based processing is that results may vary depending on the lighting conditions of the environment.

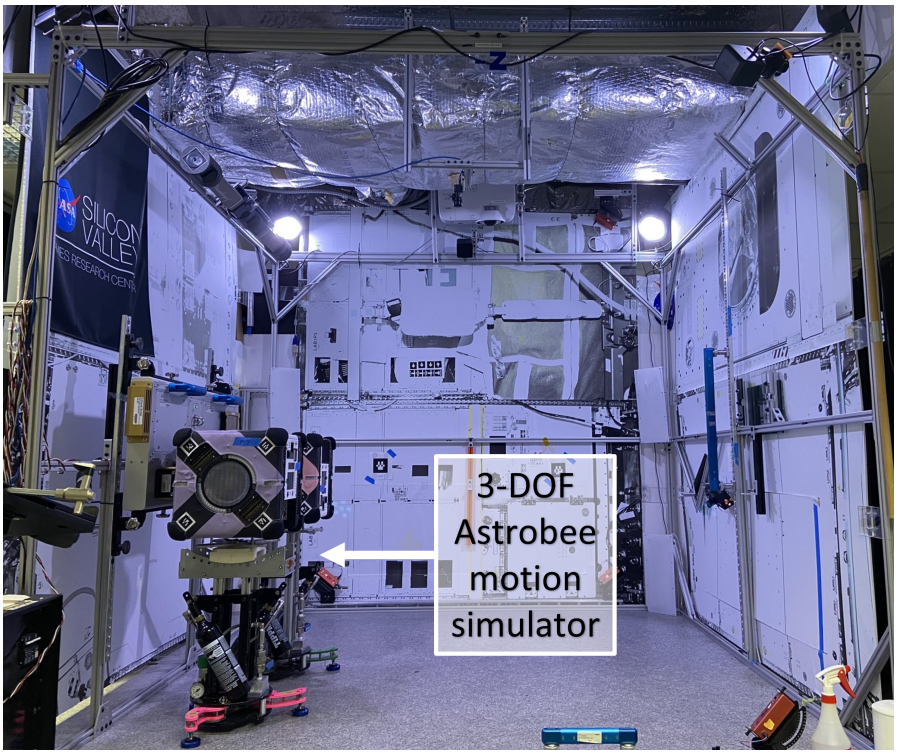


Figure 1.1: *The Granite Lab at NASA Ames Research Center simulates the frictionless environment of the ISS for testing Astrobee path planning and control. Image taken by the author.*

In light of these limitations, an unsupervised clustering strategy using depth data either from stereo reconstruction using standard RGB cameras, or directly from 3D cameras, is a fitting candidate to serve as a method for generalized anomaly detection in a space station environment such as the ISS. This thesis presents the implementation and extension of a scene change detection algorithm described in [Man+14]

within the ISAAC software framework. A modified expectation-maximization algorithm is used to cluster the data into Gaussian mixture models (GMMs) with unknown initializations for the number of Gaussians (K) in each mixture. To determine when the GMMs have reached the optimal value of K in each mixture, the minimum description length (MDL) is used as a stopping criterion. Finally, the earth mover's distance (EMD) is computed between the before and after GMMs to draw out the Gaussians contributing the highest degree of change, resulting in a set of points in the after map where the changes, if any, most likely occurred.

The following contributions have been made through this project:

1. Detecting multiple changes within a map at one time using GMM clustering of 3D depth data has been successfully demonstrated.
2. Change detection on real data collected using the Astrobee robot in the Granite Lab at NASA Ames Research Center has been demonstrated and evaluated, showing the efficacy of the method in a space habitat with localization uncertainty. The Granite Lab is used as an analog, simulating the appearance of the ISS environment and 3DOF surveying for an Astrobee robot.
3. The source code is publicly released at https://github.com/nasa/isaac/tree/master/anomaly. This is the first openly available software of which the author is aware that detects and identifies scene changes using GMM clustering on only 3D depth data.

1.1 Limitations

The objective of the change detection algorithm proposed in this work is to identify general changes within a scene, i.e., to find areas in a map in which objects have appeared or disappeared. However, relying on an unsupervised clustering algorithm for the change detection axiomatically implies that the change detection algorithm cannot detect *what* has changed in the scene, only that the scene is different in identified areas than it was at a previous time. Furthermore, since the objects themselves cannot be identified, the detected changes may be that a moved object appears as a new object in one area, and a disappeared object in the original area. A more accurate representation would label this object as "moved."

The granularity of the change detection algorithm is also limited by the resolution of the point cloud data and the size of the map. Smaller or thinner objects, such as wires, may be detected with a high resolution point cloud and smaller search area, but will be absorbed into larger clusters for large maps and lower resolution point clouds. For the purpose of this thesis, the target size of objects was around the size of the Astrobee itself. To give an idea of scale, the Astrobee is a cube approximately 0.3175 meters on each side, with a volume of about 0.03 m^3. The Granite Lab is $2m$ on each side, with a volume of $8m^3$. The Astrobee is about 0.004% of the volume of the Granite Lab.

Given these limitations, the change detection algorithm removes the major limitation of requiring manually labeled data. However, in some cases, object identification is essential and thus other algorithms should be used.

2 Theory

2.1 Related Work

The methods described in the following papers were considered for reaching the objective of detecting changes on the ISS from data provided by Astrobee free-flying robots.

2.1.1 Semantic Instance Detection

Previous work has been done to localize the Astrobee free-flying robots in microgravity by generating semantic maps offline and localizing against the map online [Mil+22]. Semantic map construction entails running a subset of images through a Structure from Motion pipeline to obtain a bundle-adjusted pose graph and localizing the images in the sparse map. Per-class object heatmaps are then generated to locate objects. To localize the robot, a graph optimization procedure is used which compares detected objects to poses in the graph. A data association algorithm is finally used to minimize the total distance between these object pairs.

This work lends itself as a foundation for change detection; in a next iteration, the location of detected objects could be compared to the previous map to determine which objects have disappeared, appeared, or possibly changed positions. However, a limiting factor is labeling data for object detection. In this work, thousands of images from eight object classes were manually labeled for training. While semantic detection is critical for tasks involving object class or instance detection, such as scanning for warning labels or handrails, this strategy would not be able to detect changes in the environment due to novel object types.

2.1.2 Semantic Class Segmentation

The concept behind this approach closely relates to that behind Semantic Instance Detection, though the underlying method varies. The key difference involves assigning individual voxels to classes within the scene, rather than to instances of objects. In [Neu+11], an algorithm is described to split voxel data into segments that do not necessarily correlate to complete object classes, but to segments that optimize the change detection output. Nonetheless, 27 object classes were manually labeled for training, such as tree trunks, roads and bushes. With a voxel-based approach, it follows that the change detection algorithm may be able to attain a higher specificity than is afforded by the bounding box constraint of object detection. Nevertheless, the manual labeling of the data still presents itself as a major constraint for semantic class segmentation.

2.1.3 Comparisons of UAV RGB-D Point Clouds from 3D Image Reconstruction

While targeting low altitude, high resolution UAV images, the method outlined in [AA19] presents a pipeline of comparing point cloud data from two different periods. Initially, point clouds are generated from image-based 3D reconstruction. Once the RGB-D point clouds have been obtained, separate maps are generated and a coarse-to-fine registration process is applied to overlay the two point clouds on top of one another. Depth and grayscale difference maps are then produced and random forest classification and component connectivity analysis techniques applied to discover changed buildings.

In the context of robots detecting changes online within the ISS, maps of modules reconstructed from images have historically been patchy in areas occluded by protruding objects. The quality of these maps precludes them from being a top candidate for change detection.

Ultimately, the tuning and validation of the algorithm for the "2.5D" nature of satellite image reconstruction rather than the truly 3D environment robots experience, especially on the ISS, is why this method was not further explored. However, in future work, application of the change detection method described to ISS point cloud data could be explored to compare to the change detection algorithm chosen for this thesis.

2.1.4 Point Cloud to Base Map Comparisons

For fast 3D localization, a method of comparing updated point clouds to a base environment map has been proposed [Che+16]. 3D laser scanner-generated map data and point clouds generated by an RGB-D/stereo

camera are converted to ND (normal distribution) voxels using NDT (normal distribution transform). The voxels are sifted into three categories that are then used, in conjunction with features of ND voxels, for comparisons between the map and and measured data.

A similar method had been proposed by [Kat+19] which uses the Mahalanobis distance to compare local 3D point clouds to nearest voxels in octree-based occupancy maps of the environment. A clustering algorithm is then used to generate a list of change candidates, and outliers are removed with a random forest classifier. Classification scores and number of occurrences are used to map and report changes in real time.

While these localization techniques demonstrate real-time change detection for robotic applications, they are predicated on the availability of high quality 3D maps, which are difficult to generate using current Astrobee and ISAAC software. Additionally, new maps would need to be generated frequently to produce useful results.

2.1.5 Image Projection onto 3D Models

Another lightweight change detection algorithm is described in [PS17], which similarly detects changes between current data and a previously constructed 3D model. However, instead of reconstructing the current scene each time, an image of the current scene is back-projected onto the 3D model, and projected to a viewpoint of another current image. The projection from the model is compared to the other current image to identify differences. A multiple (4-5) keyframe sequence is used to remove ambiguities and to locate the changes in 3D space. This method presents the same issue of the quality of output depending on the quality of the available 3D map, which, as discussed, is not yet available at frequent intervals or in high quality on the ISS.

2.2 A Generalized Method for Change Detection Using Point Cloud Data

A generalized change detection mechanism is desirable for various applications in robotics and on the ISS, for instance as a first step to finding vents blocked by cargo bags. While methods such as semantic segmentation are tantamount to solutions requiring object identification, they are limited by the laborious task of labeling, a necessity that also limits the identification of unlabeled objects. By using an unsupervised learning and detection method, the only limitations to changes detected are the quality of data and portions of the algorithm that should be optimized in future iterations.

One such method is outlined in [Man+14], the paper which provides the main inspiration for this project. Using 3D point cloud depth data as input, a cluster map in the form of a Gaussian Mixture Model (GMM) is constructed using a modified Expectation-Maximization (EM) algorithm. This representation reduces the feature space from thousands of points down to several orders of magnitude fewer Gaussians, which are in turn represented by their means and weights (or proportion of the distribution). Using this condensed format, the Earth Mover's Distance algorithm can be applied to compare the maps at times t_0 and t_1. The Gaussian in the t_1 map that contributes the highest degree of change (represented by the EMD) is removed, added to a third "change" GMM, and the process is repeated until the EMD stops decreasing. The end product is a GMM that represents the areas in the map where changes have been detected in the scene.

2.2.1 Gaussian Mixture Models

A mixture model is a statistical representation of a set of data that identifies subsets within the data. Gaussian distributions are a common sub-model used in mixture models, and have been chosen for this thesis because of the ability to define a variance in every dimension independently. Specifically, they better describe oblong objects than do spheres, for example. In the context of scene change detection, a singular set of thousands of points can be modeled as a mixture of on the order of 10^0 - 10^1 Gaussians.

A Gaussian mixture model is defined as:

$$p(S^t|\mathbf{\Theta^t}) = \sum_{i=1}^{K} w_i g(S^t|\mu_i, \sigma_i) \tag{2.1}$$

where g represents the component Gaussian densities and w the weights of the Gaussians [Rey09]. A single Gaussian distribution is defined as:

$$g(S^t|\mu_i, \Sigma_i) = \frac{1}{(2\pi)^{\frac{D}{2}} |\sum_i|^{\frac{1}{2}}} \exp\{-\frac{1}{2}(S_t - \mu_i)' - \Sigma_i^{-1}(S^t - \mu_i)\} \tag{2.2}$$

with a mean vector μ_i and covariance matrix Σ_i. The weights, w_i, are normalized such that $\sum_{i=1}^{K} w_i = 1$ [Rey09]. Each Gaussian $\mathbf{\Theta^t}$ can be summarized by its parameters:

$$\mathbf{\Theta^t} = \{w_i, \mu_i, \Sigma_i\} \qquad i = 1, ..., K \tag{2.3}$$

While Gaussian mixture models offer a convenient way to summarize the point clouds, there remain two unknowns that must be solved before they can be used:

1. Which point belongs to which cluster?
2. How many clusters should be used?

For the first question, the expectation-maximization algorithm can be used. For the second, two different initialization methods have been explored.

2.2.2 Clustering using Expectation-Maximization

The objective of the expectation-maximization (EM) step is to summarize the point cloud data into a mixture of 3D Gaussians with unknown means and covariances. The EM algorithm is an iterative process that alternates between an expectation step, in which the probability that each data point belongs to each Gaussian is calculated, and the maximization step, which calculates the portion of points allocated to each Gaussian, and subsequently the updated means and covariances. The process is repeated until convergence of the log-likelihood of the model describing the data, taking into account a penalty term to discourage too many Gaussians from being used.

Given a complete data set with known points, $\mathbf{S^t}$, and the clusters to which each point belongs, $\mathbf{Z^t}$, the log-likelihood estimate is given by [FJ02]:

$$\mathcal{L}(\mathbf{S^t}, \mathbf{Z}|\mathbf{\Theta^t}) = \sum_{j=1}^{n} \sum_{i=1}^{K} z_i^j \log[w_i p(s^j|\boldsymbol{\theta}_i)] \tag{2.4}$$

However, $\mathbf{Z^t}$ is a latent variable which the EM algorithm seeks to estimate.

Expectation Step

With the "E" step, the expected value of the latent variable $\mathbf{Z^t}$ is calculated. For each data point s_j, the expected value $E(z_{ij})$ is the probability that s_j exists in cluster i [M06]:

$$E(z_{ij}) = \frac{p(z_k = 1)p(s_j|z_k = 1)}{\sum_{i=1}^{k} p(z_i = 1)p(s_j|z_i = 1)} \tag{2.5}$$

Equation 2.5 can be rewritten as:

$$E(z_{ij}) = \frac{w_i \mathcal{N}(s_j|\mu_i, \Sigma_i)}{\sum_{i=1}^{K} w_k \mathcal{N}(s_j|\mu_i, \Sigma_i)} \tag{2.6}$$

with w_i serving as the prior probability, and $E(z_{ij})$ the posterior probability once s_{ij} has been observed [Mit97].

Maximization Step

The weights of each of the Gaussians are calculated based on the point allocations, and the means and covariances are then updated [M06]:

$$m_i = \sum_j \mathrm{E}(z_{ij}) \tag{2.7}$$

$$w_i = \frac{m_i}{m}, \qquad \sum_{i=1}^{K} = 1, \qquad 0 \leq w_i \leq 1 \tag{2.8}$$

$$\mu_c = \frac{1}{m_c} \sum_i \mathrm{E}(z_{ij}) \mathbf{s_i} \tag{2.9}$$

$$\sum_i = \frac{1}{m_i} \sum_j w_{ij} (\mathbf{s_i} - \mu_j)^T (\mathbf{s}_i - \mu_j) \tag{2.10}$$

using the updated means from Equations 2.9 in 2.10, and the total sum of weights for m in Equation 2.8. The log-likelihood of the given model is defined by:

$$\mathcal{L}(\mathbf{S_t}|\mathbf{w}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{j=1}^{N} \log \left(\sum_{i=1}^{K} w_i \mathcal{N}(\mathbf{s_i^t}|\boldsymbol{\mu_i}, \boldsymbol{\Sigma_i}) \right) \tag{2.11}$$

The cyclical process of updating the expected values of the weights and the parameters of the model is continued until convergence of Equation 2.11.

Initialization and Stopping

The traditional expectation-maximization algorithm begins with a pre-determined number of Gaussians. For example, given a dataset with two distributions, the expectation-maximization algorithm aims to determine which of the two distributions to each point in the dataset belongs. However, when modeling real, complex scenes, the true number of Gaussians is arbitrary. For this algorithm, the best number of Gaussians is the smallest number that can accurately separate the objects within the scene. In other words, an object in a model may be separated into several Gaussians, but it should not be grouped with the surrounding wall or table. A fewer number of Gaussians is desired in order to simplify distance calculations later on.

In this thesis, two different methods of were investigated to determine a proper K-value:

1. Running a grid search over an expected range of Gaussians (Grid Search Method)
2. Beginning with the maximum acceptable number of Gaussians (Annihilation Method)

Grid Search

The simpler method is to scan through an array of models with different parameters: covariance type, number of components (Gaussians), and penalty term for scoring. The model with the lowest score is chosen.

The objective is to minimize the log-likelihood of the points given the model, plus a penalty term:

$$\boldsymbol{\Theta}^* = \arg\min_{\boldsymbol{\Theta}} \{ -\log \mathcal{L}(\mathbf{S}^t|\boldsymbol{\Theta}) + P(K) \}. \tag{2.12}$$

The following penalty terms were were evaluated by [Man+14], and therefore were also considered in this thesis [Man+14]:

- Minimum Description Length (MDL)[FJ02; Man+14]:

$$P(K)_{MDL} = \frac{D}{2} \sum_{k=1}^{K} \log \left(\frac{N w_k}{12} \right) + \frac{K}{2} \log \left(\frac{N}{12} \right) + \frac{K(D+1)}{2} \tag{2.13}$$

- Bayesian Information Criterion (BIC):

$$P(K)_{BIC} = 0.5 J \log(N) \tag{2.14}$$

- Akaike Information Criterion (AIC):

$$P(K)_{AIC} = 2J \tag{2.15}$$

Here, J represents the number of free parameters ($10 \times K - 1$), D is the number of parameters used to describe one Gaussian, and N is the number of observations, or points in the original set. As the models in each iteration are initialized with the ultimate number of Gaussians to be used, the performance of each model is highly dependent upon the random initialization of the means, weights, and covariances of each cluster [FJ02].

Unsupervised Learning with Annihilation

This method, described in [FJ02], is an unsupervised solution to finding the optimum number of Gaussians that best fit the model. At a high level, the algorithm begins with random initialization of a high number of Gaussians (e.g., 25), and weakly supported clusters are gradually removed, with stronger clusters absorbing leftover points.

More specifically, the initial means are randomly selected from the data, and the covariances are proportional to the identity matrix

$$\hat{\mathbf{C}}(t=0) = \sigma^2 I \tag{2.16}$$

and σ^2 is proportional to the average variances along each dimension of the data [FJ02]:

$$\sigma^2 = \frac{1}{10d}\left(\frac{1}{n}\sum_{j=1}^{n}(s^j - m)(s^j - m)^T\right) \tag{2.17}$$

The MDL criterion (Equation 2.13) was selected by [FJ02] as the penalty term for the log-likelihood cost function because it intuitively rewards models using a minimum parameter encoding; the $-\log p(\mathbf{S}^t|\mathbf{\Theta}^t)$ term directly relates to the code length of the data. Furthermore, the term

$$\frac{D}{2}\log N w_i \tag{2.18}$$

in Equation 2.13 gives the best code length (parameter mixture) for each Gaussian θ_i, as Nw_i represents the expected number of points used to create θ_i [FJ02].

Therefore, assuming Dirichlet-type priors for the weights, the probability of the weights is proportional to Equation 2.18 [FJ02]:

$$p(w_1, ...w_K) \propto \exp\{-\frac{D}{2}\sum_{k=1}^{K}\log w_k\} \tag{2.19}$$

The Dirichlet prior is a conjugate function to multinomial likelihoods such as Equation 2.11. Therefore, to minimize Equation 2.11, the M-Step can be modified to reflect the negative exponent in Equation 2.19. Less important (lower weight) clusters are eliminated by assigning them an expected weight of 0 once the sum of the weights for a cluster minus half the number of dimensions for the data becomes negative, as this violates the negative weight criterion (Equation 2.8):

$$w_k = \frac{\max\left(0, \sum_{j=1}^{N} w_i^{(j)} - \frac{D}{2}\right)}{\sum_{k=1}^{K}\max\left(0, \left(\sum_{j=1}^{N} w_k^{(j)}\right) - \frac{D}{2}\right)} \tag{2.20}$$

Starting with a large number of Gaussians helps to mitigate the random initialization problem present in the Grid Search method because local maxima of the likelihood may occur where there is a high density of Gaussians in one area of the model, and low density in other areas. With a high density everywhere, local maxima are avoided. Additionally, overlapping Gaussians are handled by the competitive nature of the modified M-Step, as the negative exponent of Equation 2.19 makes the prior unstable [FJ02].

One issue that may occur when using a high number of Gaussians at the start is that none of the clusters dominate, and thus all of the cluster weights will go to 0. Therefore, the EM algorithm is performed component-wise, or by cluster, and not for all clusters at once, as is the case with the traditional EM algorithm [FJ02].

2.2.3 Calculating the Earth Mover's Distance

The earth mover's distance (EMD), also known in mathematics as the Wasserstein metric, provides a quantitative measure of how different two distributions are from one another. As opposed to other metrics, such as the

Kullback-Leibler (KL) divergence, the earth mover's distance accounts for the metric distance between points in a distribution. For example, two given instances of a scene with a blank background may be identical except for a box translated a certain distance to the right. The KL divergence for the two scene instances will be similar regardless of if the distance the box moved was half a meter or two meters. However, the earth mover's distance accounts for the metric distance of the box moved: the further the box is moved, the larger the EMD.

Intuitively, the EMD can roughly be thought of as the amount of work required to transform one distribution into another. One distribution can be analogized as a distribution of dirt, or earth, and the other distribution as holes to be filled. In the context of Gaussian mixture models, the weights of the Gaussians themselves serve as a measurable unit for the "earth" component of the algorithm. The metric distances between the means of $\mathbf{\Theta}^{t_0}$ and $\mathbf{\Theta}^{t}$ can directly be used in the work calculation.

To find the EMD, the optimal flow (total weight moved), $\mathbf{F}$, must be found, with the following constraints [RTG00]:

$$\begin{aligned}
&\underset{\mathbf{F}}{\text{minimize}} \quad \text{Work}(\mathbf{\Theta}^{t_0}, \mathbf{\Theta}^{t}, \mathbf{F}) = \sum_{i=1}^{M}\sum_{j=1}^{N} d_{ij} f_{ij} \\
&\text{subject to} \\
&f_{ij} \geq 0, 1 \leq i \leq M, 1 \leq j \leq N, \\
&\sum_{i=1}^{M} f_{ij} \leq w_i^{t_0}, 1 \leq i \leq M, \\
&\sum_{j=1}^{N} f_{ij} \leq w_j^{t}, 1 \leq j \leq N, \\
&\sum_{i=1}^{M}\sum_{j=1}^{N} f_{ij} = \min\left(\sum_{i=1}^{M} w_i^{t_0}, \sum_{j=1}^{N} w_j^{t}\right)
\end{aligned} \tag{2.21}$$

Finding the optimal flow is a variation of the standard transportation optimization problem. The EMD, $d(\mathbf{\Theta}^{t_0}, \mathbf{\Theta}^{t})$, results from normalizing the work by the total flow:

$$d(\mathbf{\Theta}^{t_0}, \mathbf{\Theta}^{t}) = \frac{\text{Work}(\mathbf{\Theta}^{t_0}, \mathbf{\Theta}^{t}, \mathbf{F})}{\sum_i^M \sum_j^N f_{ij}}. \tag{2.22}$$

Normalization is required in order not to favor distributions with higher total flow [RTG00].

Once found, the EMD can now be used to compare two GMMs to measure the dissimilarity between the two. The flow of weights and metric distances are shown in Figure 2.1.

2.2.4 Greedy Change Detection Algorithm

The EMD is used in a greedy selection algorithm to extract the clusters from $\mathbf{\Theta}^{t}$ which contribute the highest amount of change [Man+14]. The Gaussian distribution whose removal best decreases the EMD between $\mathbf{\Theta}^{t_0}$ and $\mathbf{\Theta}^{t}$ is extracted. Extracted clusters are transferred to the final model, $\mathbf{\Pi}$, which stores detected changes. This process iterates until convergence of the EMD between any two clusters. The final model, $\mathbf{\Pi}$, can then be used to identify the points in $\mathbf{S}^{t}$ where changes occurred. This process is summarized in Algorithm 1.

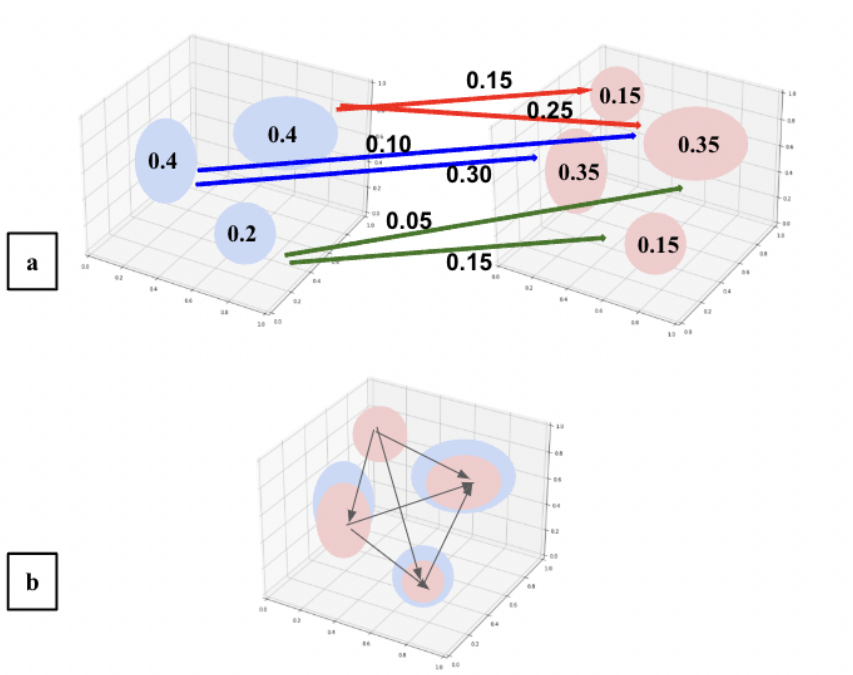


Figure 2.1: *In (a), an example of the flow of weights between the two distributions is shown. In (b), the two GMMs are algined to demonstrate how the distances between the means of two GMMs is measured.*

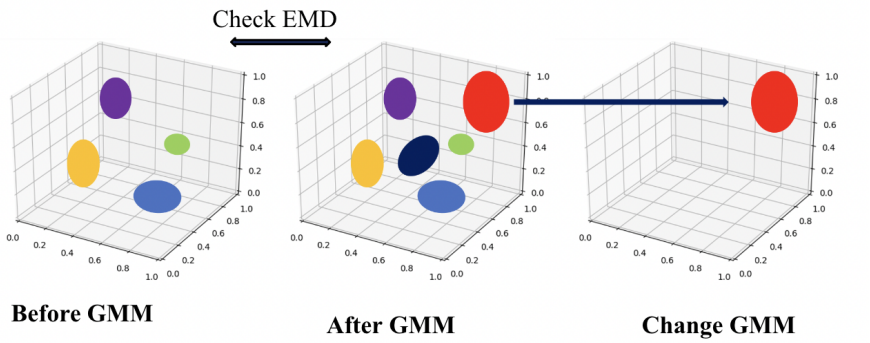


Figure 2.2: *Once the GMMs of the point clouds have been obtained, the Gaussians contributing the most to the EMD value are iteratatively removed from the "after" GMM and moved to the final "change" GMM.*

Algorithm 1 Change Detection Algorithm

$\Pi \leftarrow 0$
$EMD_{old} \leftarrow EMD(\Gamma, \Theta)$
$\Theta_{best} \leftarrow \Theta$
while $EMD_{old} > EMD_{lowest}$ **do**
  for $gaussian$ in Θ **do**
    $gaussian_{best} \leftarrow 0$
    $\Theta_{temp} \leftarrow \Theta - gaussian$
    if $EMD_{new} < EMD_{lowest}$ **then**
      $EMD_{lowest} \leftarrow EMD_{new}$
      $gaussian_{best} \leftarrow gaussian$
      $\Theta_{best} \leftarrow \Theta$
    end if
  end for
  $\Theta \leftarrow \Theta_{best}$
  $\Pi \leftarrow \Pi + gaussian_{best}$
end while
return Π

3 Methods

A change detection algorithm based on the work described by Manso et al. [Man+14] was implemented and tested with both artificially generated data and real-world data collected by an Astrobee robot in an ISS-like environment. Starting with two 3D point clouds, each representing the scene at times t_0 and t, the data is preprocessed using the Statistical Outlier Removal and Voxel Grid downsampling filters from the Point Cloud Library [RC11].

To simplify the calculations required to compare the two point clouds, which may consist of tens of thousands of points, the data is summarized into a few (on the order of 5 to 25) Gaussian clusters. Clustering is performed using an unsupervised variation of the expectation-maximization algorithm. Once this step is completed, the two resultant Gaussian Mixture Models (GMMs) can then be compared directly.

The change detection algorithm can be summarized as an iterative process in which the two GMMs are compared, and the Gaussian from the t GMM, $\mathbf{\Theta}^t$, that contributes the highest degree of change is moved to the "change" GMM, $\mathbf{\Pi}$. To compare the two Gaussians, the earth mover's distance (EMD) calculation is used. Once removing a GMM from $\mathbf{\Theta}^t$ no longer improves, or stops decreasing, the EMD, the algorithm is complete and the finalized $\mathbf{\Pi}$ is provided as the output. This process is summarized by Figure 3.1.

3.1 Data

This section describes the collection and processing of the input data to the change detection algorithm.

3.1.1 Point Cloud Data

Point cloud data was selected as the input for the change detection algorithm as it is impervious to changes in lighting, i.e., the results of the algorithm will not change regardless of the shadows or lighting conditions between two instances. However, most algorithms, including the one this thesis is based on ([Man+14]), also incorporate RGB data as input to provide extra robustness to the data. This was not feasible using Astrobee, as the Picoflex HazCam only provides poor quality black and white data. Use of the color cameras (SciCam and NavCam) would require registration of depth sensor data to the RGB data which is captured on other parts of the Astrobee robot.

The Astrobee flight software uses the ROS software framework to capture the real-time sensor data, including that from the depth and image sensors. The depth data is captured as a series of sensor_msgs::PointCloud2 messages. There are two variations of sensor_msgs::PointCloud2: ordered and unordered. The data is captured as ordered data, i.e., the array of coordinates is recorded in the two dimensions of the resolution of the depth sensor. To simplify format converting, the ordered sensor_msgs::PointCloud2 messages are converted to arrays of shape [1,N,3], where N represents the total number of points collected and 3 is the dimensionality of the collected data (x,y,z).

3.1.2 Data Collection

The data can be segmented into four categories: artificially generated, single frames from the 3D depth camera, and stereo image and ground truth reconstructions of the entire map.

Artificial Data

As a simple proof-of-concept and validation check for testing, data was artificially generated to test how well various parts of the algorithm performed. The objective of testing using artificial data is twofold:

1. Determine how well the clustering algorithm can detect the number, means, covariances and weights of the underlying Gaussians

2. Test the change detection algorithm to determine how well it can identify which and how many underlying Gaussians changed between the two instances

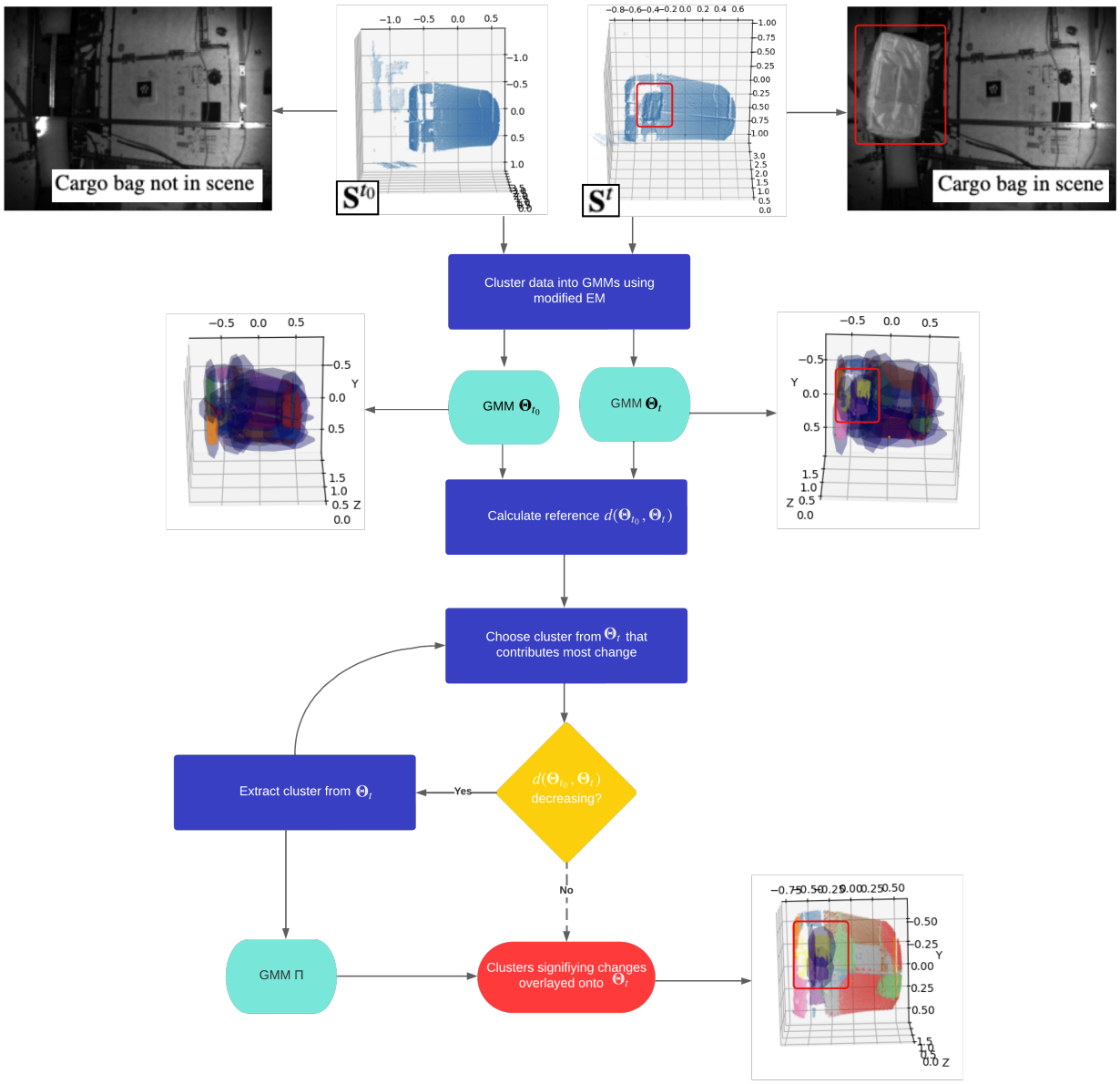


Figure 3.1: *The change detection algorithm first clusters input point clouds* $\mathbf{S}^{t_0}$ *and* $\mathbf{S}^{t}$ *into Gaussian distributions using a modified Expectation-Maximization algorithm. Changes are detected using the earth mover's distance (EMD) to compare the Gaussian distributions.*

Initially, one "scene" or point cloud was created by generating four 3D Gaussian clusters with specified means and covariances using the NumPy library's random multivariate function. In a second time instance, three more clusters were added in a similar fashion. The points from each set were then concatenated to form two NumPy arrays of (x,y,z) coordinates. The initial point clouds used for testing are shown in Figure 3.2. Eventually, a method of generating random point clouds with a specified number of starting Gaussians, number to add, and number to remove, was added to the software pipeline. This was implemented in order to test performance across a variety of Gaussian numbers and placements.

Laboratory Data

For the latter three categories (single frames and the two different mapping methods), the data was collected in the Granite Lab at NASA Ames Research Center. The lab simulates the ISS environment of the Astrobee, complete with a docking station, ISS-like walls, and a nearly frictionless surface (a massive granite table) for

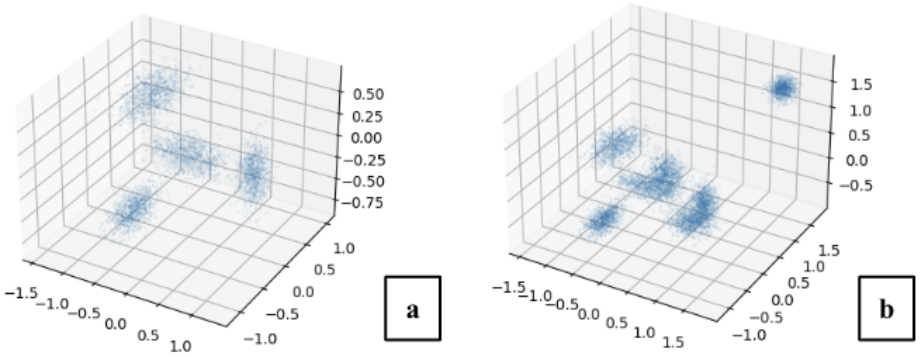


Figure 3.2: *Artificial point clouds were generated by concatenating predetermined multivariate Gaussians. The point cloud on the right, (b), is the same as (a), but with points drawn from three additional Gaussians, including some overlapping of points.*

the Astrobee to glide on in two dimensions. The Astrobee robot makes use of two impellers that allow it to control its motion [Col+16], and is set on a predetermined survey route. During these preprogrammed surveys, Astrobee captures scene information via three separate cameras: the HazCam, NavCam, and SciCam. While the NavCam and SciCam each capture high quality RGB image data, the HazCam captures both lower quality black-and-white images and 3D depth data. The HazCam onboard is a PMD Picoflex that obtains depth data via time-of-flight measurements.

To collect data, two pre-programmed routines can be performed: geometry surveys and panoramic surveys. In both cases, a path is drawn out in a software simulation of either the Granite Lab or the ISS. For the Granite Lab, with the geometric survey, the Astrobee glides a fixed distance away from the wall, forming a rectangular path. Similarly, if a panoramic survey is performed, the Astrobee rotates from the center to capture data from the scene. For data collection, the geometry survey was arbitrarily chosen.

An initial two surveys were conducted to capture scenes with and without a prop cargo bag in the corner of the Granite Lab. Five more surveys were later performed to obtain multiple changes in the scene, with different combinations of the objects being moved between scenes. There were three objects: another Astrobee, an approximately 0.5 meter cargo bag, and a similarly sized tool box (shown in the lab in Figure 3.7). A description of the five scenes is captured in Figure 3.6. These objects were chosen as they represent the approximate size of objects of interest on the ISS, such as real cargo bags that are used to store supplies. An example of a scene on the ISS is shown in Figure 3.3.

3.1.3 Single Frames (Raw Point Cloud Data)

Raw point cloud data from the Hazcam depth sensor was used as input data for initial testing with real world data. Initially, two instances of the Granite Lab were captured: one with a cargo bag in the corner, and one without. Various ROS tools (*rqt* and *rosbag*) were then used to scan the output bag file (the zipped sensor data) for the timestamp at which the cargo bag was detected by the depth sensor at time t. The same was then done to find the timestamp at which the Astrobee was at the same location in the scene, time t_0. The bags were cut at these timestamps, and a script was created to extract the Hazcam black and white images (for visual inspection) and sensor_msgs::PointCloud2 messages at the start of these cropped bag files. The output was then two point clouds, $\mathbf{S}^{t_0}$ and $\mathbf{S}^{t}$.

3.1.4 Volumetric Reconstructed Maps

While single frame depth data has the benefit of requiring no processing after collection, and therefore may be a good candidate for running the algorithm online, the raw data requires registration in order to compare changes over entire maps. Registration methods such as Iterative Closest Point (ICP) were considered [ZYD22].

Figure 3.3: *This image was taken on the International Space Station, showing the relative size of the cargo bags and other objects used onboard. This image is used by courtesy of ESA's online ISS Virtual Tour: https://esamultimedia.esa.int/multimedia/virtual-tour-iss/.*

However, included in the ISAAC software package is a method of reconstructing the scene map based on input depth data and the Voxblox library processing [Ole+17]. Bundle adjustments and ground-truth localization adjustments improve map fidelity. The pipeline produces meshes with vertices that can be extracted and reformated as point clouds.

Example outputs are shown in the final column of Figure 3.6, as well as a close-up in 3.4. In the latter, the data collected beyond the scope of the lab can be observed as two pillars on either side of the open rectangle (the lab). Nevertheless, the data within the bounds of the lab accurately depict the scene and all objects can easily be observed by visual inspection.

3.1.5 Ground Truth Localized Maps

An alternative option to the volumetric reconstructed maps generated is ground truth localized data [Sou+22]. This option uses raw point cloud data as well as accompanying ground truth transformation data provided as ROS messages from the bag file. A reduced number of point clouds is chosen (for example, approximately every hundredth message depending on the frequency of the depth images collected), and their corresponding transforms are found (according to timestamp). The ground truth localized point clouds are then concatenated into a single NumPy array.

The output of this method has less fidelity and more noise than that produced by volumetric reconstruction. In Figure 3.5, an entire wall can be seen beyond the lab. However, this artefact can be ameliorated by setting scanning boundaries during the surveys for online evaluation, or by offline cropping. The largest benefit offered by localizing against raw data is that it is more straight-forward to run online than volumetric reconstruction.

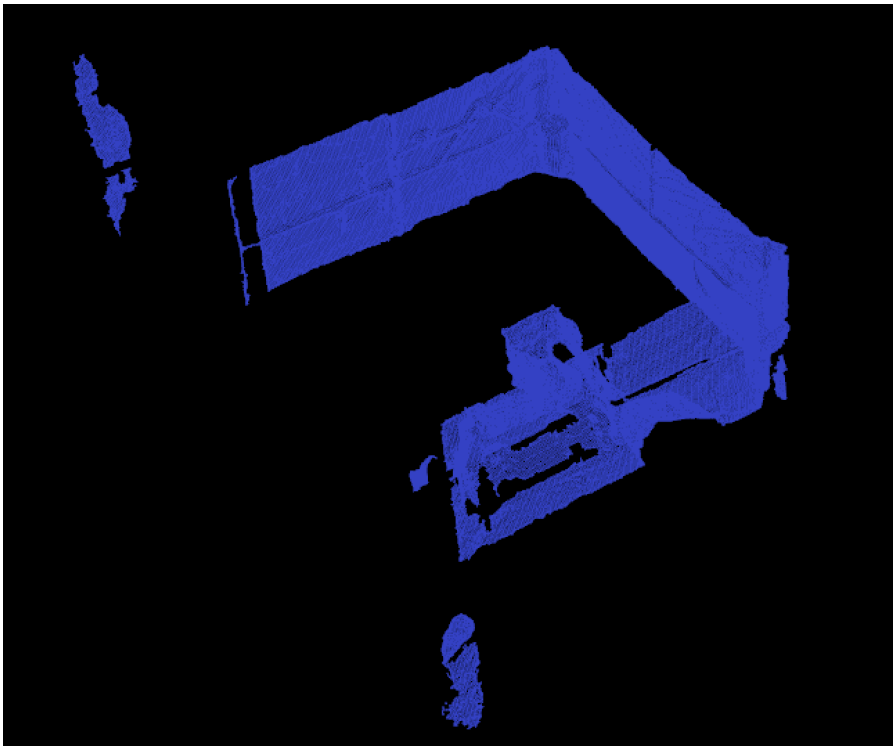


Figure 3.4: *Volumetric reconstruction is used to produce accurate meshes of the scene. This is an isometric view of one of the maps produced with Geometry Survey 2 in Figure 3.6. The nearest surface shows the backside of the Astrobee, docking station, and cargo bag near the floor. The two columns of points on either side of the map are background noise beyond the walls of the lab.*

Figure 3.5: *A second method for reconstruction considered was ground truth localization of raw point cloud data. The end result is less accurate than volumetric reconstructed maps, but is easier to obtain. The input bag file data is the same as that shown in Figure 3.4*

.

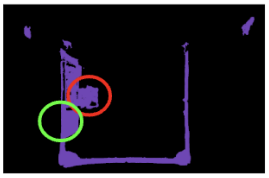
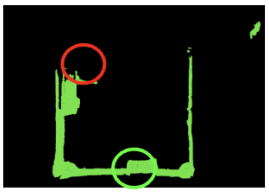
Run	No. Objects	Astrobee	Cargo Bag	Black box	Volumetric Reconstructed Maps
1	2	In corner near dock	On back wall (high, middle)	-	
2	2	Docked (away from edge)	On docking wall	-	
3	2	Docked, near edge (not properly captured)	On back wall (low, right)	-	
4	3	Docked (near edge)	Back wall (high, left)	Opp. dock, low corner	
5	0	-	-	-	

Figure 3.6: *Shown is the top-view of volumetric reconstructed maps of the Granite Lab. Encircled in red is the Astrobee, green is the cargo bag, and blue is the gear box.*

3.1.6 Preprocessing of data

Preprocessing is necessary to filter out extraneous points, and to reduce the size of the point cloud while retaining all pertinent information. For the purpose of this change detection algorithm, the Point Cloud Library was chosen to filter and handle the data.

Point Cloud Library

The Point Cloud Library is used as an intermediate between ROS sensor_msgs::PointCloud2 data and NumPy arrays. Other point cloud formats were considered, but Point Cloud Library in particular features two filters also used by [Man+14]: Statistical Outlier Removal and Voxel Grid Filter. Data in PCL format can also be easily converted to NumPy arrays and vice versa. A Python wrapper, Python-PCL, was used for implementation.

Statistical Outlier Removal

During data collection, stray points are captured by the depth sensor. It is possible that outliers far from other data can absorb Gaussians during the expectation-maximization clustering step, i.e., are assigned their own

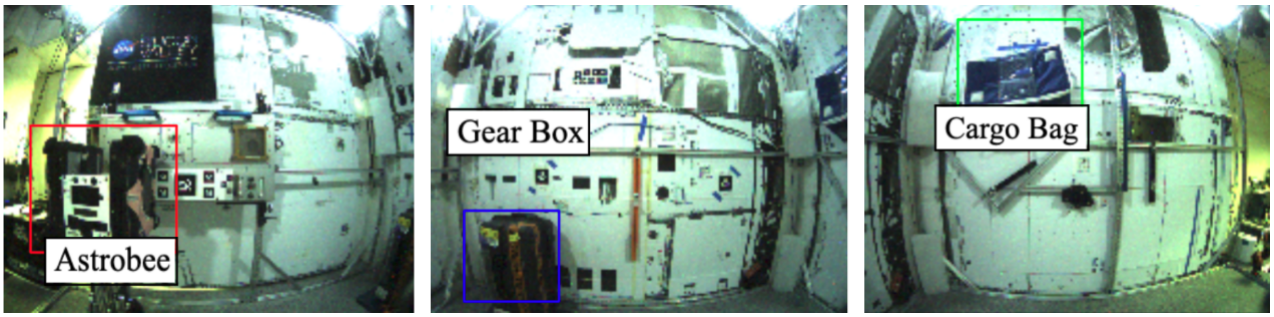


Figure 3.7: *The Astrobee scene object is placed near the edge of the lab scene, and the Gear Box and Cargo Bag are placed in corners of the lab scene. The placement of the Astrobee robot at the edge of the scene. Point clouds produced from this vantage are subject to significant noise.*

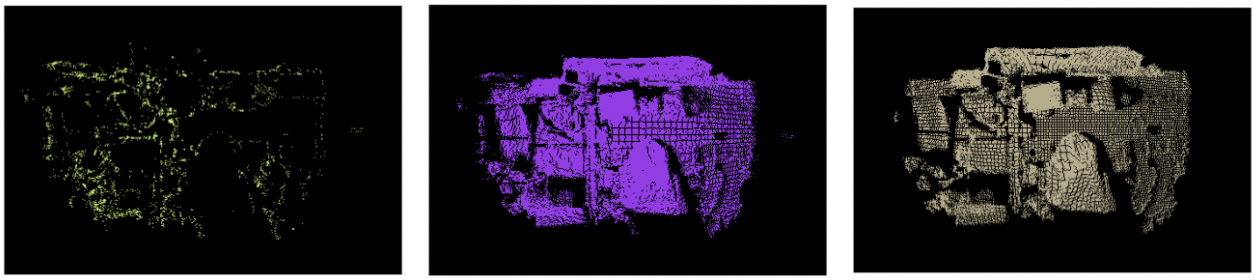


Figure 3.8: *A point cloud from the ISS is shown in the center. On the left are the outliers removed from the point cloud, and on the right, the point cloud after filtration.*

clusters. This can lead to excess Gaussians, which diminish the fidelity of the model and the efficacy of the change detection algorithm.

The statistical outlier removal filter analyzes the neighborhood of points for every point, removing points that are more than a specifed standard deviation value away from the neighborhood's mean [Man+14]. The results of the filter applied to a single raw point cloud taken from an Astrobee survey of the ISS is shown in Figure 3.8.

Voxel Grid Down-sampling Filter

A single point cloud captured from the Picoflex camera contains thousands of points. When concatenated with other point clouds for map reconstruction, the concatenated point cloud can contain tens of thousands of points. Data reduction can be advantageous for the expectation-maximization algorithm, which predicts the cluster class of each individual point. However, the down-sampling filter should not remove *too* much information, and therefore reduce the precision of the point cloud in representing objects within the scene. Therefore, a leaf size of 1 cm was chosen for the voxel grid filter.

3.1.7 Segmentation into Mixture of Gaussians

As described in Section 2, the expectation-maximization algorithm presents a chicken-and-egg type problem: the model parameters are initially unknown and estimated via the algorithm, but the model needs to be initialized with a static number of Gaussians.

Two methods of optimization were explored: grid search and a modified expecatation maximization algorithm. A stopping criterion is also necessary to determine when the log-likelihood has sufficiently converged.

Scanning Method

To implement the grid search method, the scikit-learn Python package was used [Sci]. Scikit-learn features a GaussianMixture class that implements the expectation-maximization algorithm to fit the input point cloud

and predict the class (cluster) of each point. While this method is susceptible to singularities (earlier discussed in regards to statistical outlier removal), it is the fastest algorithm for clustering [**scikit-learn**].

Scikit-learn also offers a class for parameter tuning: GridSearchCV. A scoring criterion (e.g., Bayesian Information Criterion, Akaike Information Criterion, or Minimum Description Length), range of allowable numbers of Gaussians, and types of covariances are specified as parameters to search for. Using, for example, these three criteria, four covariance types (spherical, tied, diagonal and full), and a component range of [3,8], 144 models are found (see Figure 3.9) The model with the lowest score is chosen.

Modified Expectation-Maximization

After implementing the grid search method, a more elegant solution was investigated. The GMM begins with the maximum number of Gaussians (with a default K of 25). The Gaussians in the model are initialized with random means, covariances, and proportion of points (weights). Starting with a high entropy environment, combined with random initialization, allows the GMM to self-anneal slowly and avoid trapping itself in sub-optimal local minima of the log-likelihood. [FJ02] After initialization, the standard EM algorithm is run, except for a modification to the M step which forces Gaussians with low weights to self-annihilate.

An open-source Python implementation of the algorithm described by [FJ02], based on MATLAB code written by the authors, was used [Abr20].

3.1.8 Earth Mover's Distance Calculation

To calculate the earth mover's distance between two Gaussian mixture models, a measure of weight (in a physical sense) of the clusters and a metric distance framework are required. The cluster weights of the GMMs can directly be used for the earth weight required to be moved in order to transform one distribution into the other. A distance matrix between all means in $\mathbf{\Theta}_{t_0}$ and $\mathbf{\Theta}_t$ (such that connecting the means creates a bipartite graph) is calculated using the straight-forward distance formula.

The optimal flow, or minimum amount of total weight required to be moved, is then calculated in order to find the least amount of work needed such that the distributions are the same. As this is a variation of the famous transportation problem, in which warehouse supply needs to be moved to the customers with the least amount of work possible, the optimal flow is modeled as such. Considering one distribution to be the warehouse supply, the other distribution to be the warehouse demand, and the distance between the means as the cost of transportation, the constraints and variables can be fed to a linear programmer. For this implementation, the Python linear programmer, PuLP was used. The final EMD is the work output by the PuLP caluclation, normalized by the smaller total sum of weights of the two distributions. Since in this implementation all weights sum to one, the EMD is equivalent to the minimal work.

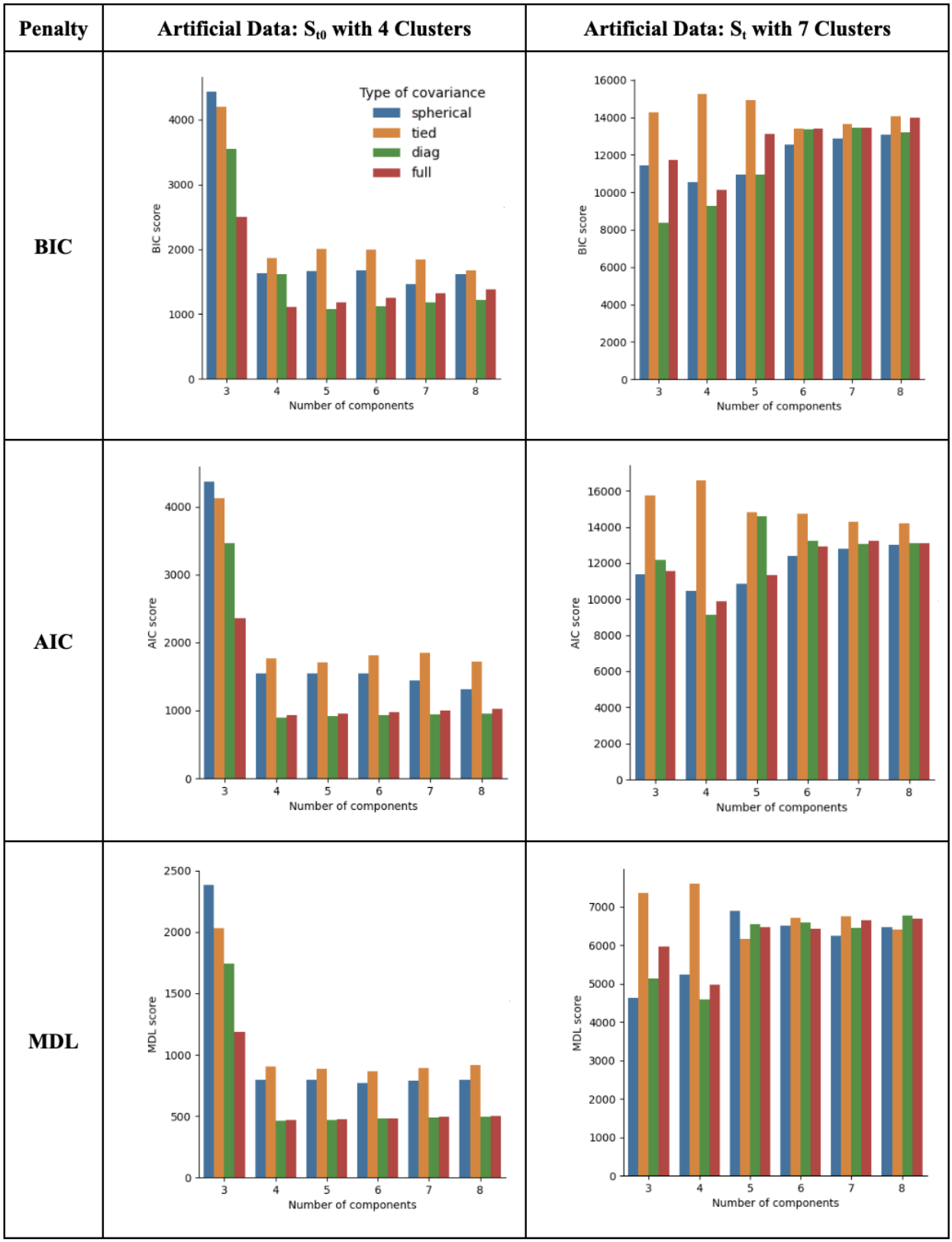


Figure 3.9: *With the scanning method, a set of models using an array of various stopping criteria, covariance types, and number of Gaussians were found. In the left column, the input was an artificial point cloud consisting of points drawn from four Gaussians. The input for the column on the right was the same, except with three added Gaussians. The model (represented by a single bar in each of these charts) with the lowest score is selected for each point cloud.*

4 Results and Discussion

The pipeline was tested in stages, starting with artificial data and ending with running on the entire reconstructed map of the Granite Lab.

4.1 Artificially Generated Data

Initial testing with generated data served as a foundational check to confirm that each stage of the pipeline was working as expected. In one instance, the "before" point cloud was generated using a multivariate normal random sampler function in Python, with 4000 points drawn from four 3D Gaussians of varying means and covariances. For the "after" point cloud, 3000 more points were drawn from three more Gaussians, with some Gaussians intersecting others. Two of the three added Gaussians intersect with existing Gaussians, while a third is clearly separated (Figure 4.1: center GMM, in the upper right corner). The final results obtained are as follows:

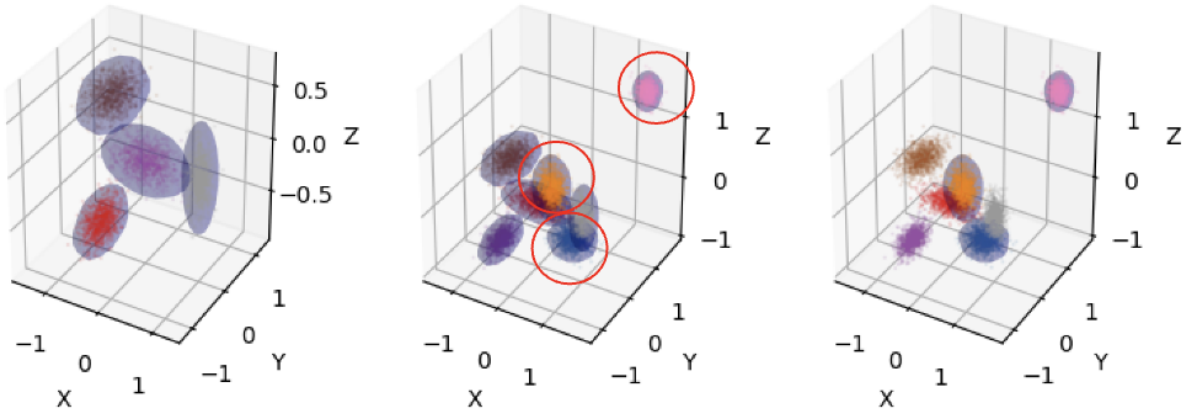


Figure 4.1: *The leftmost GMM, $\mathbf{\Theta}_{t_0}$, correctly identifies the four underlying Gaussians. Similarly, the middle GMM, $\mathbf{\Theta}_t$, correctly clusters the original four, and the three extra, as well. The change GMM or the right, $\mathbf{\Theta}_t$ identifies all three added Gaussians as differences between $\mathbf{\Theta}_{t_0}$ and $\mathbf{\Theta}_t$.*

4.1.1 Grid Search

Initial clustering of the input point clouds was performed using the grid search method described in previous sections. While clustering of the four Gaussians shown in the leftmost GMM of Figure 4.1 was consistently correct, the clustering of the seven Gaussians (shown in middle of Figure 4.1) was not. Figure 4.2 shows example clustering estimations of the seven-Gaussian point cloud using the grid search method. A range of (3,9] Gaussians was specified. In these figures, diagonal covariance and the Bayesian Information Criterion for stopping were used. Experimental results for the grid search scores are shown in Figure 3.9. In the search shown, the lowest score for the 4-Gaussian point cloud was correctly obtained using the MDL criterion with a diagonal cluster using four Gaussians. However, the 7-Gaussian point cloud was less accurate: while the lowest score was again obtained using the MDL criterion and a diagonal covariance, the lowest number of Gaussians (3) was also selected.

As previously discussed, while the grid search method is comparatively fast and simple to implement, it suffers from heavy dependency upon initialization and falling into local maxima at the parameter boundaries [FJ02]. For this reason, the grid search method was abandoned in favor of the modified expectation-maximization algorithm proposed and implemented by [FJ02].

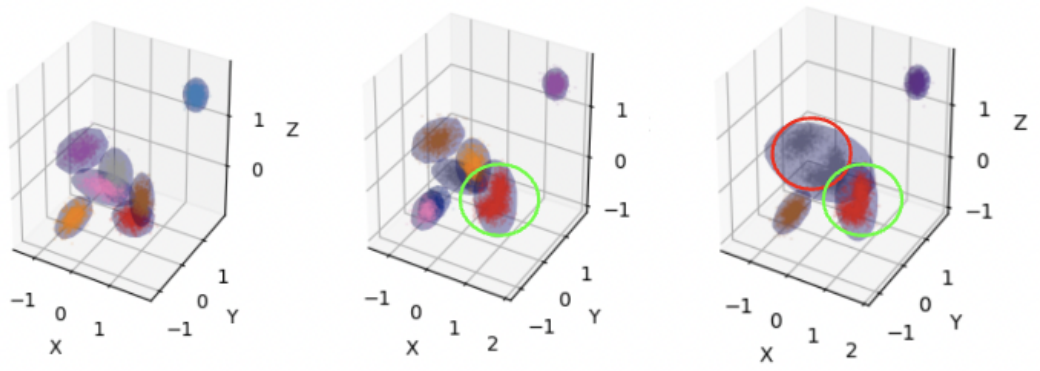


Figure 4.2: *Using the grid search method, clustering results were inconsistent, as it is highly dependent on initialization. The right two images show that the estimated clustering of the seven underlying Gaussians (shown in the center of Figure 4.1) were only six and four Gaussians respectively; points from several Gaussians were lumped together (highlighted in the circles). Here, a range of (3,9] Gaussians was used.*

4.1.2 Modified Expectation-Maximization

Results using the modified expectation-maximization are shown in Figure 4.1. Each of the Gaussians in the before and after point clouds were properly detected by the Gaussian Mixture Modeling clustering algorithm, without having knowledge of the correct number of Gaussians in each point cloud a priori. Furthermore, the three added Gaussians were correctly identified by the change detection algorithm, which also has no indication of the number of changes to search for beforehand. Results are much more consistent, producing the same results nearly every time.

4.2 Single Shots from 3D Camera

The initial test with real world data (albeit, in a lab setting) was with single shots taken from the HazCam during two different surveys. In the first survey, there were no objects in the scene. In the second survey, a cargo bag was added to the corner of the lab, opposite the docking station. Single shots were obtained by lining up the robot poses in each iteration exactly and finding the corresponding timestamps within the collected bag files. In Figure 4.3, the cargo bag is shown as two Gaussian clusters, which were detected by the change detection algorithm as changes in the scene. The change Gaussians are mapped over the "after" point cloud, $\mathbf{\Theta}_t$, which is now color coded according to each point's predicted Gaussian cluster.

Mirroring the results obtained using the artificial data, superior clustering was obtained using the modified expectation-maximization algorithm. As shown in Figure 4.4, the grid search method sometimes defaulted to the minimum number of specified Gaussians in the range. This prevented objects changed in the scene from being identifiable as they were clustered together with larger areas, such as the walls.

4.3 Volumetric Reconstructed Map

Entire maps of the scene were constructed using a pipeline within the ISAAC framework that processes depth data using Voxblox and bundle adjustments to improve localization[Ole+17]. Figure 4.5 shows two different scene changes using these volumetric reconstructed maps as input. In the first scenario, two objects (an Astrobee and a cargo bag) were added on the same wall. Like with the individual frame point clouds, a high number of Gaussians is used for each volumetric reconstructed point cloud. Despite the presence of the Astrobee docking station, which may be clustered with added scene objects in false positive clustering results, the Astrobee robot and cargo bag are both identified scene changes.

Three objects added to the scene was then tested, as illustrated in the right half of Figure 4.5. In this scene, a gear box, cargo bag, and Astrobee were placed on separate walls, with the Astrobee placed on the

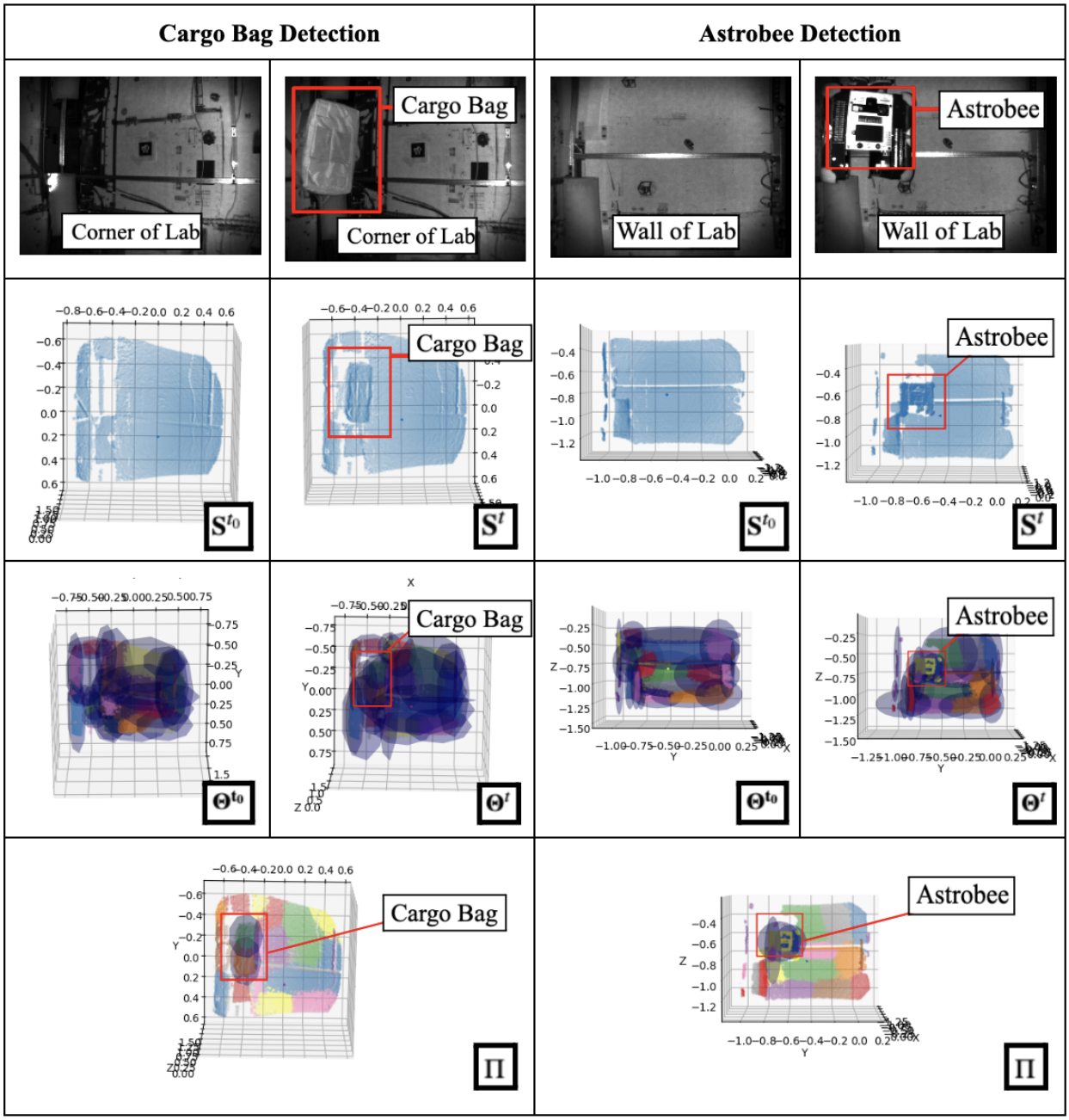


Figure 4.3: *A cargo bag and Astrobee robot are detected within the Granite Lab using single depth frames taken at the same location at different times. (Left Column) Cargo bag change detection (Right Column) Astrobee robot change detection (1^{st} Row) Depth images (2^{nd} Row) Filtered point clouds (3^{rd} Row) GMM clustering with modified-EM (4^{th} Row) Detected changes.*

docking station. The docking station is placed near the edge of the lab, which places the Astrobee next to significant noise from the background. The position of the Astrobee can be seen in 3.7. Possibly due to this background noise, the Astrobee was not detected as a change. However, in this test, the gear box and cargo bag were both correctly isolated and identified. Furthermore, there were no incorrect identifications.

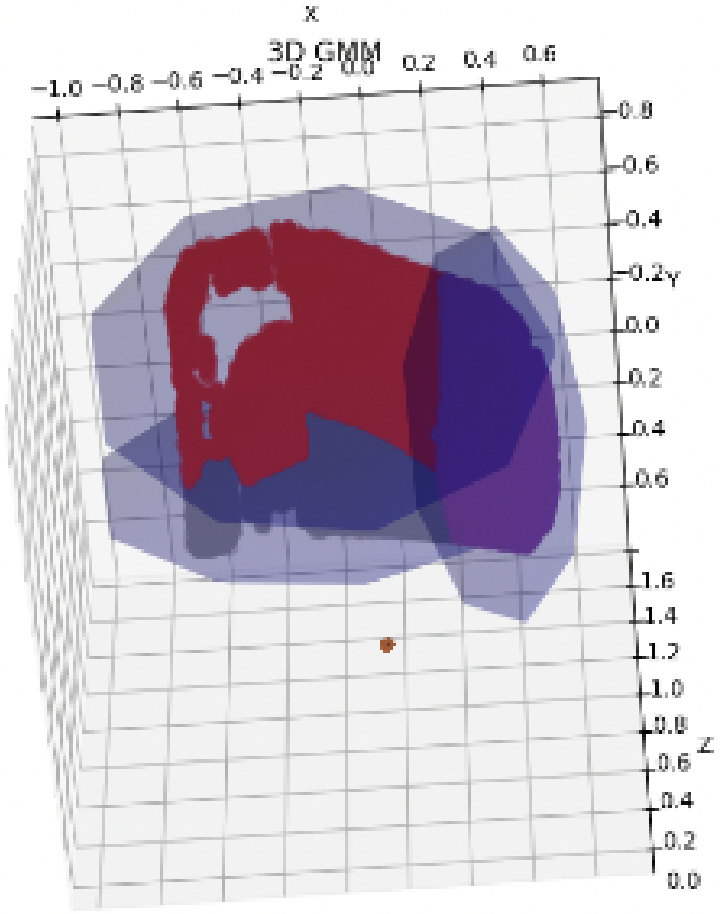


Figure 4.4: *Clustering using the grid search algorithm is susceptible to defaulting to the parameter boundaries; here, the minimum number of Gaussians is selected, and the object (cargo bag) and wall are clustered together.*

4.4 Ground Truth Localized Data

Because the ground truth data for this method (rotation and orientation of the Astrobee) is derived from a previously registered map, the resulting maps were much noisier and less accurate than the same scenes depicted with volumetric reconstruction. The noise from one map (Run 2 of Figure 3.6) is shown in Figure 4.7. The algorithm was run on the map with and without statistical outlier filtering (Figure 4.6). Without filtering, extraneous random points can be seen floating in space, in addition to additional frames captured outside of the lab. This noise overwhelms the algorithm and hides any true changes. Results are improved with filtering, but misaligned walls are still registered as changes. These results can be improved using iterative closest point (ICP) registration to better align the walls after localization.

4.5 Detecting Disappearances

The detection of disappearances is not directly addressed in the algorithm described by [Man+14] [Wel+17]. The proposed change detection algorithm works by removing *extra* Gaussians from the *after* ($\boldsymbol{\Theta}_t$) GMM. Therefore, as-is, the change detection algorithm cannot find Gaussians associated to object disappearances in the *after* GMM.

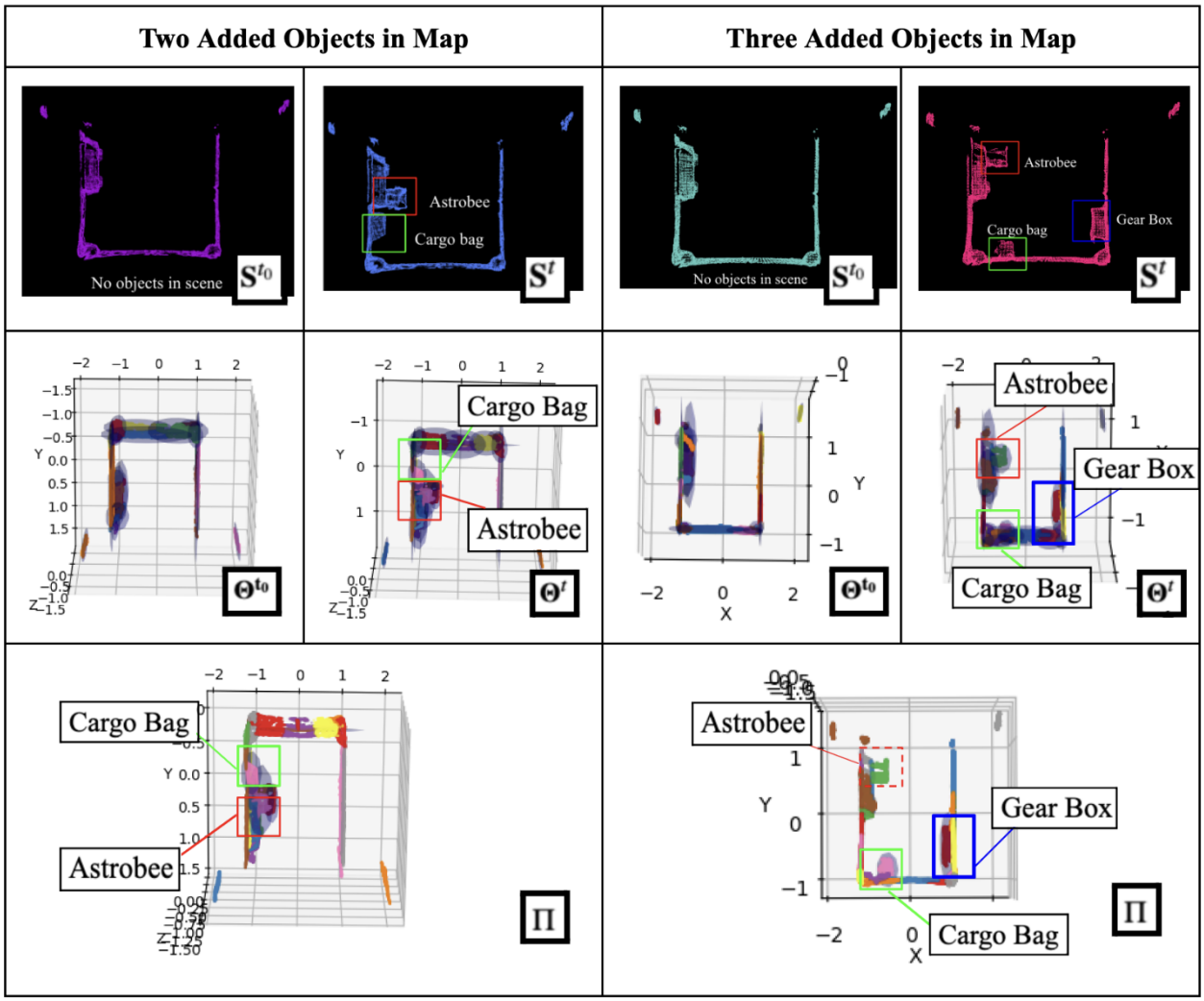


Figure 4.5: *Two different scene comparisons are shown: on the left column, an Astrobee and cargo bag are added to the scene, and on the right, an Astrobee is also added.* $\mathbf{\Theta}^{t_0}$ *and* $\mathbf{\Theta}^{t_0}$ *is shown for both scenarios. In the 2-Object scenario, both of the changes were correctly identified. However, in the 3-Object scenario, an Astrobee located in front of unfiltered background noise was not detected.*

4.5.1 Detecting disappearances only

Given that the detection of appearances can be summarized as finding the extra Gaussians contributing to the EMD metric, the detection of disappearances can be re-framed as the detection of appearances from ($\mathbf{\Theta}_t$) to ($\mathbf{\Theta}_{t_0}$). Then, the addition to the change detection algorithm is trivial; using the existing clusterings, the "before" and "after" GMMs can be switched to retrieve the additions going backwards in time.

4.5.2 Simultaneous detection of appearances and disappearances

Detecting multiple appearances and disappearances using GMM clustering produces more unstable results than performing a search for one or the other. While it is possible to detect an appearance or disappearance relatively far away from each other in the map, the closer one is to another, the more difficult both are to detect. Since the algorithm is not able to identify the objects themselves and only changes in the point distributions, a new object in the vicinity of a disappeared object may appear as the same object from the point of view of the algorithm. An example of this occurence is depicted in Figure 4.8.

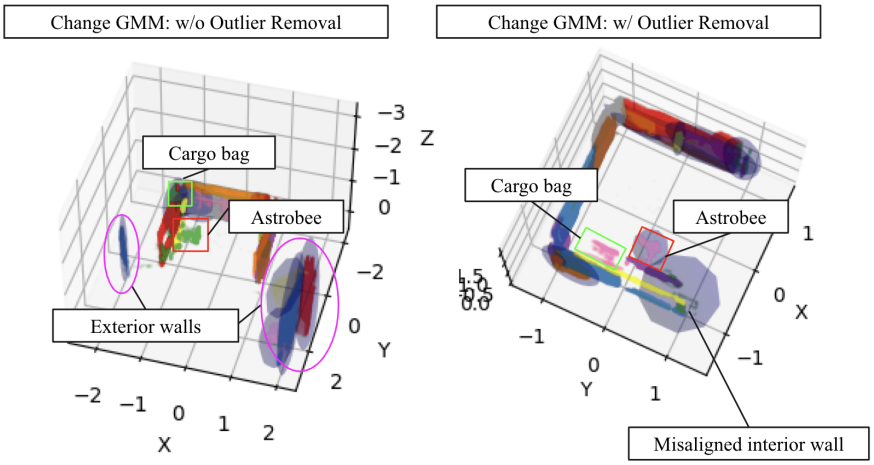


Figure 4.6: *With only the downsampling filter (left), the ground truth localized map contains too much noise for the algorithm to detect any changes correctly. Even with the statistical outlier removal filter (right), misaligned walls are not consistent between the before and after maps and are therefore detected as changes.*

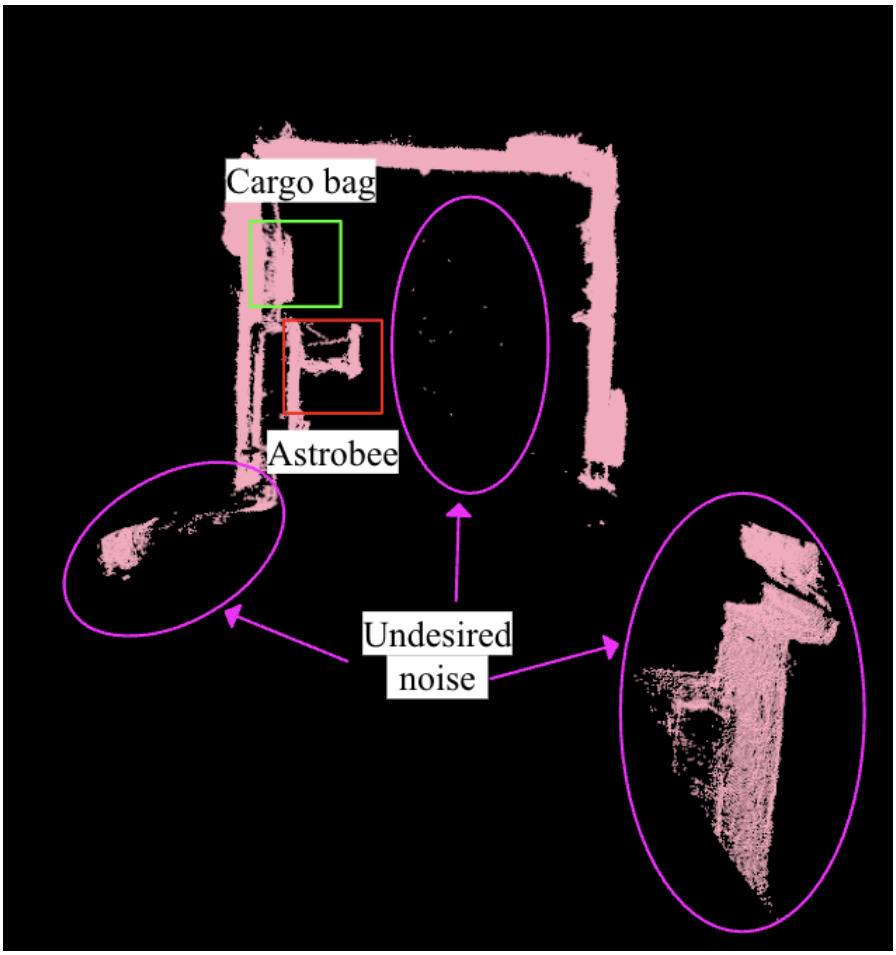


Figure 4.7: *Maps generated using ground truth localized data tend to have much more noise and inaccurate localization than the same maps generated using volumetric reconstruction.*

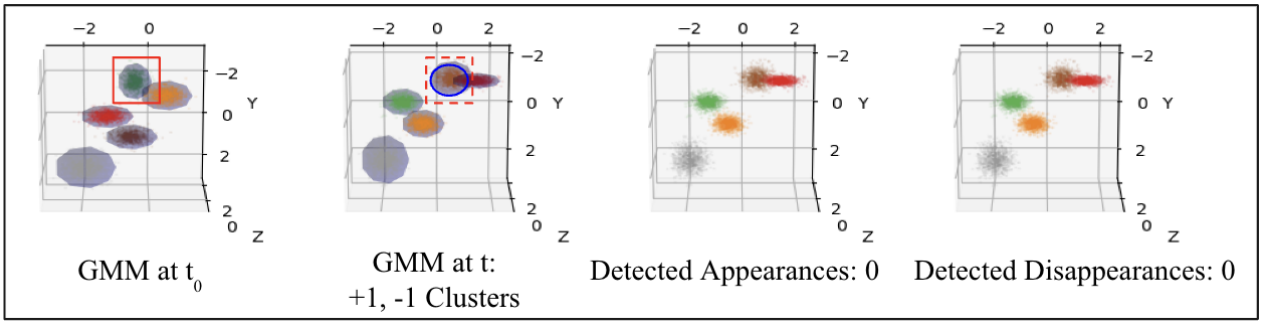


Figure 4.8: *Highlighted in red is a Gaussian that disappears in the "After" GMM. However, another, similar Gaussian appears in the same location (highlighted in blue). This results in neither being detected.*

4.6 Performance Metrics

Metrics were gathered to compare performance across different artificial and real data scenarios and to highlight the strengths and weaknesses of the algorithm. For each test, a true positive was taken to be a Gaussian associated with or overlapping a change in the map. The F1 metric, a balance between precision and recall, in this case is the most descriptive metric of the data as the number of true negatives greatly outnumbers the number of true or false positives every time; in other words, even if the algorithm detects no correct changes, the output can still have a high accuracy. On the other hand, the rate of true changes being detected as changes (recall) and detected changes being true changes (precision) is of greater interest in the context of this algorithm.

4.6.1 Artificial Data

To test the performance of the pipeline in different conditions under the ideal case, a test suite over three different variables was performed. Initial artificial point clouds were created with five starting Gaussians with varying degrees of overlap (low or medium), number of changes (one or two), and type (appearance, disappearance, or both). For low overlap, the Gaussians means were generated with a range from [-4,4] and covariances with a range of [0.01, 0.1]. For medium overlap, the mean range was reduced to [-2,2]. Trials were run 10 times for each set of test conditions. Figure 4.9 shows the resulting accuracy, precision, recall and F1 metrics. Every condition with only one type of change had an F1 score of approximately 80%. In comparison, the trials that involved one appearance and one disappearance had a significantly lower F1 of 52%. This can be explained by the fact that in the case that a Gaussian is added near the location of another removed Gaussian, the algorithm cannot distinguish between these points.

4.6.2 Real World Data

The same performance metrics were evaluated for each of the three different types of real world data: single frame depth images, volumetric reconstructed maps, and ground truth localized data. Single frame data using the cargo bag produced the highest recall of 1.0. This may be because this input type has the largest object-to-search boundary ratio, so finding the detected object is simplest in this case. The volumetric maps had the highest precision (81%). While the reason why is less straightforward, it could be that there are fewer false positives at the edges of the volumetric reconstructed map than there are at the borders of the single frame depth data, since the single frames of the before and after scenes must found manually, which introduces alignment error. From the F1 scores, and by simple visual inspection of Figure 4.10, the ground truth localized data performed the worst. However, this was to be expected as the level of noise and localization errors is significantly higher than present in the other two input types.

4.7 False Positives and Negatives

False positives (wrongfully identified Gaussians where no changes exist) are sometimes detected near the boundaries of the depth data. One simple explanation for many of these misidentifications is that the boundaries are

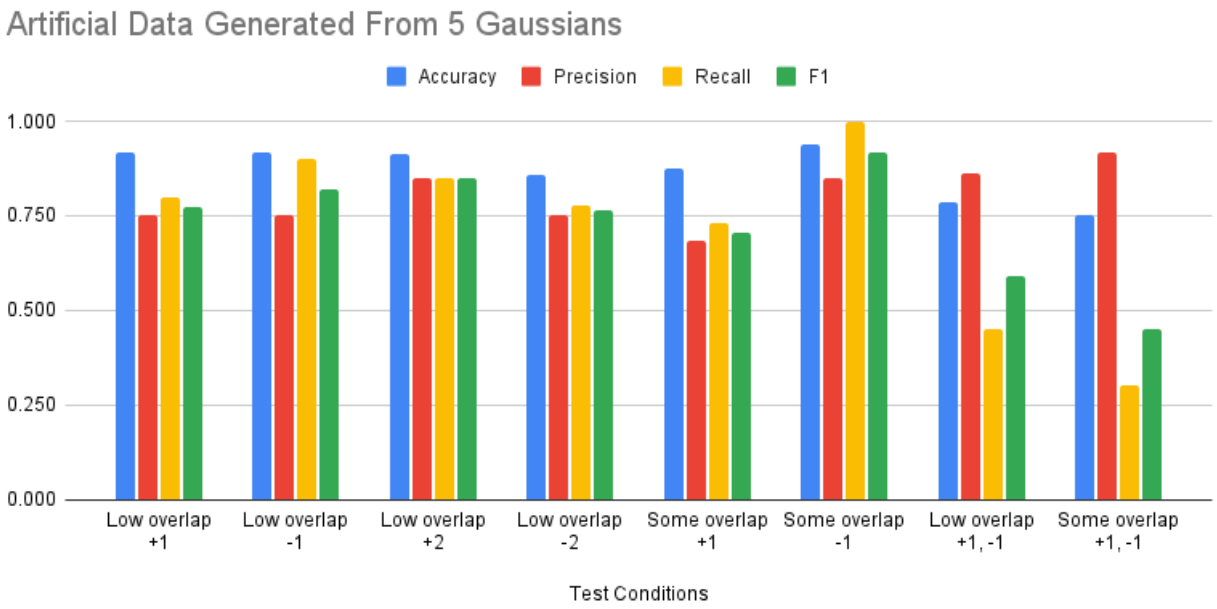


Figure 4.9: *Different conditions were tested with random artificial data drawn from 5 initial Gaussians: low overlap of Gaussians (wider range of initialization), some overlap, different numbers of added (+1, +2) and disappeared (-1, -2) Gaussians, and mixing appearances and disappearances (-1,+1).*

not perfectly aligned. For example, in the case of the single frame data, the edges must be perfectly aligned, or else any changes in the surface at the edge may be detected as a change. Figure 4.11 exemplifies this issue. While the intended changes to be identified were the Astrobee and the cargo bag, differences at the border near the cargo bag may result in clusters in that region not being properly identified (false positive of the boundary, or negative of the cargo bag).

False positives of disappearances when only appearances were introduced, and vice versa, were also evaluated. Ideally, the number of identified clusters should always be zero for either case (and thus the precision and recall will always be zero). However, in testing, some false positives were detected. The cargo bag had the highest accuracy of approximately 94%, followed by the volumetric reconstructed maps, 87%, and finally the ground truth localized data with an accuracy of 82%. The reduction of false positives can be removed by improving the accuracy of the entire model, as discussed in the Future Work section.

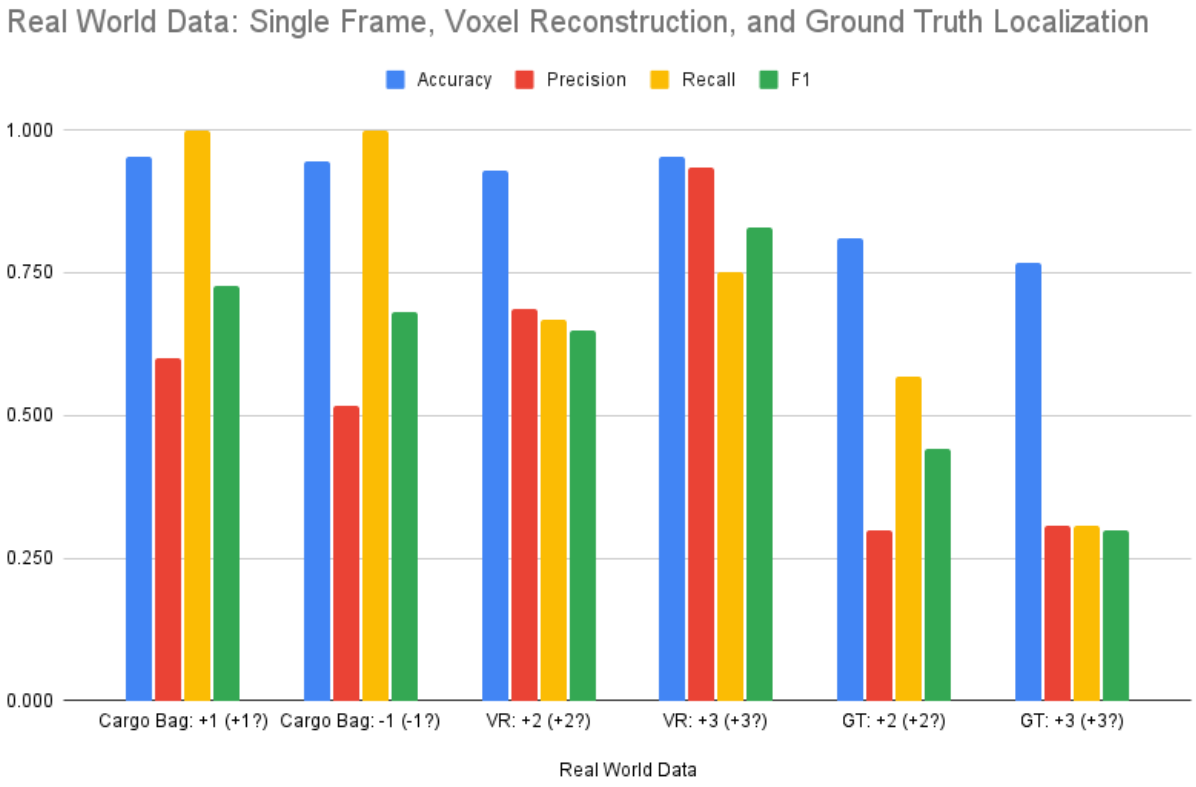


Figure 4.10: *The three types of real world data (single depth images, volumetric reconstructed maps, and ground truth localized data) were all tested. Ground truth localized data performed significantly worse than the other types of data.*

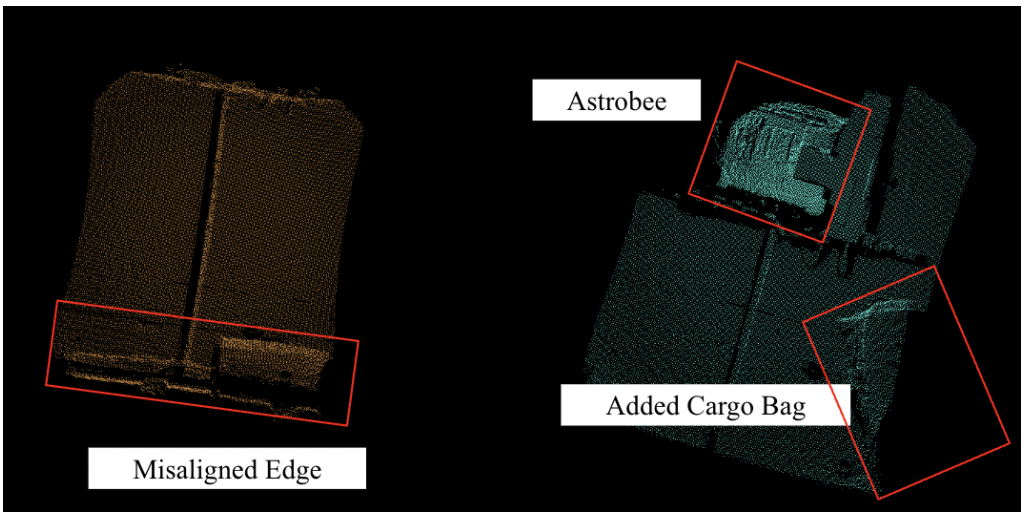


Figure 4.11: *Single frames captured from the depth sensor must be perfectly aligned, or else data at the boundary may be unintentionally counted as a change.*

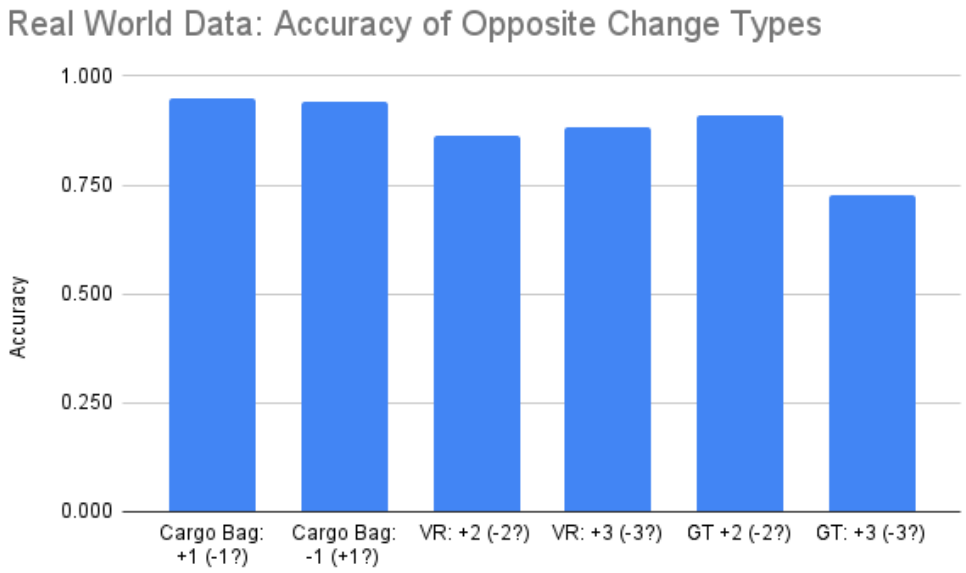


Figure 4.12: *The accuracy of detected disappearances with only appearances (and vice versa) was also tested. In the ideal case, each trial should show 0 detections for an accuracy of 1.*

5 Conclusion

The change detection pipeline offers a method for robots to detect multiple changes in their scene without dependence on manually labeled data. This offers the flexibility to detect changes in environments to which new objects may be introduced that have never been seen or labeled before. In addition, the use of point cloud data as input for the change detection algorithm is robust in relation to changes in lighting conditions between instances. Performance of the change detection algorithm in the ideal case, using artificial data, was shown to be effective at clustering and identifying added and removed changes. Across trials tested with one or two changes of the same kind (appearances or disappearances) had an F1 score of 80%. However, having several added and removed changes together in the scene was a more challenging case for the algorithm to accurately identify changes, with an F1 score of 52%. When the algorithm was applied to single images of real world depth sensor data with one change in the scene, the algorithm accurately identified the new (or removed) object with an F1 score of 70%. Applying the algorithm to reconstructed maps also produced promising results, especially when using volumetric reconstructed maps with two and three added objects (F1 of 74%). This experiment demonstrated that detection of multiple changed objects in a map of the scene, rather than just a single section, is possible. However, more work is necessary to refine the algorithm to improve accuracy.

5.1 Future Work

Accuracy can be improved by averaging results over several runs, and preprocessing the data to accurately portray the same exact scene (i.e., with complete overlap). Clustering can also be refined by tuning the max K value when handling large maps such that the algorithm doesn't settle on the highest number of Gaussians each time. Additionally, careful attention is needed to ensure that changed objects near the boundaries are isolated from the noise beyond the boundary, an issue illustrated in the Astrobee image of Figure 3.7. Testing with ISS data should also be completed once modules within the ISS are remapped to portray scene changes.

References

[AA19] A. Asokan and J. Anitha. Change Detection Techniques for Remote Sensing Applications: A Survey. *Earth Science Informatics* **12** (2019), 143–160.

[Abr20] G. Abreu. *gmm-mml*. Oct. 2020.

[Che+16] B. Chen et al. Building Change Detection with RGB-D Map Generated from UAV Images. *Neurocomputing* **208** (2016), 350–364.

[Col+16] B. Coltin et al. "Localization From Visual Landmarks on a Free-Flying Robot". *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2016, pp. 4377–4382.

[CP16] W. H. Chun and N. Papanikolopoulos. "Robot Surveillance and Security". *Springer Handbook of Robotics*. Springer International Publishing, 2016, pp. 1605–1626. DOI: 10.1007/978-3-319-32552-1_61. URL: https://doi.org/10.1007/978-3-319-32552-1_61.

[Der+21] E. Derner et al. Change detection using weighted features for image-based localization. *Robotics and Autonomous Systems* **135** (2021), 103676. ISSN: 0921-8890. DOI: https://doi.org/10.1016/j.robot.2020.103676. URL: https://www.sciencedirect.com/science/article/pii/S0921889020305169.

[FJ02] M. Figueiredo and A. Jain. Unsupervised Learning of Finite Mixture Models. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **24**.3 (2002), 381–396. DOI: 10.1109/34.990138.

[Kat+19] U. Katsura et al. "Spatial Change Detection Using Voxel Classification by Normal Distributions Transform". *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, pp. 2953–2959.

[M06] C. M. *Pattern Recognition and Machine Learning*. en. 1st ed. Information Science and Statistics. New York, NY: Springer, Aug. 2006.

[Man+14] L. J. Manso et al. "A Novel Robust Scene Change Detection Algorithm for Autonomous Robots Using Mixtures of Gaussians". *International Journal of Advanced Robotic Systems*. Vol. 11. 2. SAGE Publications, Jan. 2014, p. 18. DOI: 10.5772/57360. URL: https://doi.org/10.5772/57360.

[Mil+22] I. D. Miller et al. "Robust Semantic Mapping and Localization on a Free-Flying Robot in Microgravity". *2022 International Conference on Robotics and Automation (ICRA)*. 2022, pp. 4121–4127. DOI: 10.1109/ICRA46639.2022.9811862.

[Mit97] T. Mitchell. *Machine Learning*. en. McGraw-Hill series in computer science. New York, NY: McGraw-Hill Professional, Mar. 1997.

[Neu+11] B. Neuman et al. "Segmentation-Based Online Change Detection for Mobile Robots". *2011 IEEE International Conference on Robotics and Automation*. 2011, pp. 5427–5434. DOI: 10.1109/ICRA.2011.5980532.

[Ole+17] H. Oleynikova et al. "Voxblox: Incremental 3D Euclidean Signed Distance Fields for On-Board MAV Planning". *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2017.

[PS17] E. Palazzolo and C. Stachniss. "Change Detection in 3D Models Based on Camera Images". *9th Workshop on Planning, Perception and Navigation for Intelligent Vehicles at the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*. 2017.

[RC11] R. B. Rusu and S. Cousins. "3D is Here: Point Cloud Library (PCL)". *2011 IEEE International Conference on Robotics and Automation*. 2011, pp. 1–4.

[Rey09] D. Reynolds. "Gaussian Mixture Models". *Encyclopedia of Biometrics*. Springer US, 2009, pp. 659–663. DOI: 10.1007/978-0-387-73003-5_196. URL: https://doi.org/10.1007/978-0-387-73003-5_196.

[RTG00] Y. Rubner, C. Tomasi, and L. J. Guibas. The Earth Mover's Distance as a Metric for Image Retrieval. *International Journal of Computer Vision* **40**.2 (2000), 99.

[Sci] Scikitlearn. *2.1. Gaussian Mixture Models*. URL: https://scikit-learn.org/stable/modules/mixture.html.

[Sou+22] R. Soussan et al. "AstroLoc: An Efficient and Robust Localizer for a Free-flying Robot". *2022 International Conference on Robotics and Automation (ICRA)*. 2022, pp. 4106–4112. DOI: 10.1109/ICRA46639.2022.9811919.

[Wel+17] L. Wellhausen et al. "Reliable real-time change detection and mapping for 3D LiDARs". *2017 IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR)*. 2017, pp. 81–87. DOI: 10.1109/SSRR.2017.8088144.

[ZYD22] J. Zhang, Y. Yao, and B. Deng. Fast and Robust Iterative Closest Point. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **44**.7 (2022), 3450–3466. DOI: 10.1109/TPAMI.2021.3054619.

DEPARTMENT OF MECHANICS AND MARITIME SCIENCES
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY