applied sciences

MDPI

Article

Information Rich Voxel Grid for Use in Heterogeneous Multi-Agent Robotics

Steven Balding *, Amadou Gning, Yongqiang Cheng and Jamshed Iqbal *

School of Computer Science, Faculty of Science and Engineering, University of Hull, Hull HU6 7RX, UK
* Correspondence: s.balding-2013@hull.ac.uk (S.B.); j.iqbal@hull.ac.uk (J.I.)

Abstract: Robotic agents are now ubiquitous in both home and work environments; moreover, the degree of task complexity they can undertake is also increasing exponentially. Now that advanced robotic agents are commonplace, the question for utilisation becomes how to enable collaboration of these agents, and indeed, many have considered this over the last decade. If we can leverage the heterogeneous capabilities of multiple agents, not only can we achieve more complex tasks, but we can better position the agents in more chaotic environments and compensate for lacking systems in less sophisticated agents. Environments such as search and rescue, agriculture, autonomous vehicles, and robotic maintenance are just a few examples of complex domains that can leverage collaborative robotics. If the use of a robotic agent is fruitful, the question should be: How can we provide a world state and environment mapping, combined with a communication method, that will allow these robotic agents to freely communicate? Moreover, how can this be decentralised such that agents can be introduced to new and existing environments already understood by other agents? The key problem that is faced is the communication method; however, when looking deeper we also need to consider how the change of an environment is mapped while considering that there are multiple differing sensors. To this end, we present the voxel grid approach for use in a decentralised robotic colony. To validate this, results are presented to show how the single-agent and multiagent systems compare.

Keywords: SLAM; robotic colony; robotic communication; decentralised robotic communication; heterogeneous robotic vision; voxel grid; environment mapping

Citation: Balding, S.; Gning, A.; Cheng, Y.; Iqbal, J. Information Rich Voxel Grid for Use in Heterogeneous Multi-Agent Robotics. *Appl. Sci.* **2023**, *13*, 5065. https://doi.org/10.3390/app13085065

Academic Editors: Luis Payá, Oscar Reinoso García and Helder Jesus Araújo

Received: 2 March 2023
Revised: 29 March 2023
Accepted: 12 April 2023
Published: 18 April 2023

1. Introduction

Modern robotics is a complex field that now permeates almost all areas of work, our home life, and certainly the research community. Indeed, homes contain robotic vacuums, smart cameras that can scan packages or people at the front door, and even fridges that can determine the contents and optimise the temperature setting [1–6]. Of course, this is complemented through industry, from manufacturing, often thought to be the most common place for robotics; however, offerings now come from the municipality to agriculture, military to humanitarian aid. With this growing prevalence in robotics, there is an opportunity to explore how the already ubiquitous set of robotic agents can be combined within a domain. Moreover, proper utilisation of already existing agents would forgo the need to invest time in new agents. For example, in a domestic setting, there may be a robotic vacuum in the house; the movement and vacuum function of the agent is unique, but if there is already a CCTV (closed-circuit television) unit in the house, could this not be used to help navigate the vacuum agent? Furthermore, if the vacuum agent has inbuilt LiDAR (light detection and ranging), would it not be good to pass this more detailed floor plan layout to the CCTV, in order to complete its understanding of the environment? Obviously, this is not an unsolvable problem, as much research is published on just this topic; however, if the agents are not manufactured by the same company or are not connected to the same central server, it could be impractical to take this simple problem to a real conclusion.

A similar narrative can be found in humanitarian efforts. Imagine a dynamic, ever-changing environment too dangerous for humans to navigate or assess. This scenario could arise from various natural disasters, such as an earthquake resulting in a collapsed building. In such situations, a robotic agent could be employed to map the area, identify objects or crucial points within, monitor changes over time, and, ultimately, interact with the environment to offer assistance.

However, can we realistically expect a single agent to be specifically designed for such a rare occurrence? Moreover, even if multiple agents are deployed, the environment may not remain static or offer easily identifiable reference points. Additionally, in this critical scenario, should we assume the presence of a pre-established centralised coordinator or server infrastructure? It becomes evident that these impractical expectations in a complex environment are neither feasible nor realistic.

Thus, the challenge lies in advancing collaborative robotics to capitalise on the significant progress already achieved in the field. In this paper, an approach is presented that utilises a voxel representation of the environment, and multiple agents utilising that voxel grid in order to better estimate the agent's pose and understand the environment.

The main contribution of this paper is the design of a voxel grid that incorporates environmental information, making it suitable for multi-agent systems. The key aspects of the design are as follows:

1. *Voxel grid that shows persistent probability over time*—this is a 3D representation of an environment where each voxel corresponds to a small volume in space. The grid is used to model the occupancy probability of each voxel over time, indicating the likelihood of a particular voxel being occupied by an object or obstacle. The persistent probability over time refers to the ability of the grid to maintain the occupancy probabilities of voxels over a period, even as new observations or data are added. This means that the grid can be updated with new information, allowing it to adapt to changes in the environment.
2. *Voxel grid that preserves information over time and contains detailed information*—this is a grid that preserves detailed information about the environment and tracks changes in that environment over time. The voxel grid can store information such as occupancy probabilities, surface normals, color, and texture, allowing for a more accurate representation of the environment. Additionally, the voxel grid can be updated with new information over time, allowing it to adapt to changes in the environment.
3. *A voxel grid that can be integrated into a multiagent swarm*—this is a 3D representation of the environment that can be used by multiple agents in a coordinated manner. The ability to integrate a voxel grid into a multiagent swarm is useful in a wide range of applications, such as in robotics, where multiple agents may be required to work together to accomplish a task.
4. *A voxel grid that can accommodate multiple sensor types*—this is a 3D representation of the environment that can be constructed by integrating data from multiple sensors. The key benefit of this approach is that by utilising a voxel grid, the information can be represented in a uniform manner regardless of the sensor type.

In this paper, a solution to these issues is proposed through a framework to utilise a decentralised robotic colony for environment mapping.

This paper is set out as follows: Section 2 considers the state of the art; Section 3 closely examines the proposed solution; and Section 4, finally, includes results and discussion.

2. Materials and Methods

2.1. The Problem and Why We Need Framework

Multiagent systems are most effective in diverse, dynamic, or even chaotic environments. A successful example is [2], but a more rigid communication architecture would require complex setup and adjustment of the communication method when introducing a new type of agent.

Consider a scenario involving a rescue robot capable of removing rubble and a drone surveying the environment to find survivors and provide a path for the other agent. This real-world problem highlights the benefits of multiagent systems and the challenges encountered with traditional SLAM (simultaneous localisation and mapping) methods. The issues identified in the previous section are informative in this context.

The first problem arises when the drone maps a "safe" path for the ground agent but the latter cannot locate the same landmarks due to object obfuscation, sensor noise, or bad orientation. The dynamic and chaotic nature of the environment exacerbates this issue, as the ground agent may not observe the landmark correctly, or the landmark may no longer exist.

This problem is further compounded by the fact that the state space in SLAM is fully correlated to robot pose and environment landmarks [7], leading to issues with full observability

To the best of our knowledge, there has not been a direct comparison of our approach presented in this paper with existing methods. Therefore, to provide a fair evaluation of our contribution, we include a summary. The summary is of recent work to identify areas where our approach offers novelty and improvement. Table 1 lists recent contributions in the relevant fields and the challenges they faced.

Table 1. Summary of recent work in the field with key findings and challenges.

Title	Methodology	Key Findings	Challenges
A Versatile and Efficient Stereo Visual Inertial SLAM System [8]	Stereo visual-inertial SLAM system	Outperformed state-of-the-art SLAM systems on several datasets while being computationally efficient.	Difficulty in handling featureless environments and robustness to lighting changes.
ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual–Inertial, and Multi-Map SLAM [9]	Visual–inertial, and multi-map SLAM system.	Achieved state-of-the-art performance on several datasets and provided a unified SLAM framework.	Difficulty in handling large-scale environments and varying camera configurations.
Real-time LiDAR Odometry and Mapping on Variable Terrain using Surface Normal Voxel Representation [10]	LiDAR odometry and mapping using surface normal voxel representation.	Improved accuracy of LiDAR odometry and mapping on variable terrain compared to traditional approaches.	Difficulty in handling dynamic environments and varying LiDAR configurations.
A Hierarchical Point Cloud Registration Method Based on Coarse-to-Fine Voxelisation [11]	Hierarchical voxel-based point cloud registration method.	Improved accuracy and robustness of point cloud registration compared to existing methods.	Limited scalability to large-scale point clouds and difficulty in handling featureless areas.
Voxel Map for Visual SLAM [12]	Voxel hashing-based monocular visual-inertial SLAM system.	Achieved state-of-the-art performance on several datasets and improved robustness to camera motion and occlusions.	Difficulty in handling large-scale environments and varying lighting conditions.
Voxel-Based Representation Learning for Place Recognition Based on 3D Point Cloud [13]	Place recognition based on voxel grid.	The proposed voxel-based representation learning method is effective for place recognition using 3D point clouds. The method outperforms state-of-the art methods in terms of accuracy and robustness to changes in viewpoint and lighting conditions. The method can be trained end-to-end using only raw 3D point clouds without the need for additional preprocessing steps.	Limited to place recognition tasks and may not be suitable for other applications that require a more detailed understanding of the 3D environment, such as object recognition or scene reconstruction.

2.2. SLAM

SLAM [14], originally developed by Hugh Durrant-Whyte and John J. Leonard, based on earlier work by Smith, Self, and Cheeseman [15,16] and the first paper to declare the complete process of SLAM, is the method by which an agent maps an environment whilst understanding its position within the newly mapped environment. The importance of this is obvious within the context of a robotic colony mapping an environment. This methodology is so robust and ubiquitous in modern robotics that it is a natural backbone for the robotic agents used within this thesis. The basic idea behind SLAM is broken down into these simplified steps:

- Gather sensor data from the agent;
- Collect sensor data into a map of the environment;
- Track agent movement at time T via the use of onboard odometry;
- Detect features/landmarks within the environment;
- Track features/landmarks at time T;
- Move agent;
- Update predicted position based on odometry;
- Update position based on features/landmarks accounting for uncertainty in both odometry and sensor readings.

This paper considers multiagent SLAM. A good review can be seen in [17]. A more recent overview of the state of the art still separates the problem into single and multiagent SLAM. The evaluation is of particular interest, with the difference between traditional landmark-based SLAM, the graph method, and the Monte Carlo.

There are examples of multiagent SLAM [4,18,19] that look toward solving the multiagent problem. Although these examples are proven to work, the identified gap in this paper is in decentralised communication between agents; furthermore, multiagent SLAM remains to be generalised between heterogeneous agents. With multiagent SLAM, the pose of the agent, the feature detection, and the use of odometry are rigid. This should not present an issue; however, with the voxel grid approach proposed in this paper, certain advantages can be conferred. Firstly, the choice of feature detection will be taken out of the equation and moved to the specific agents. In this paper, the use of SURF (Speeded Up Robust Features) and frame-to-frame alignment will be explored; however, this is not core to the philosophy of the voxel grid approach—more pertinent is that the environment is aligned and there is a closed loop described in 3D space.

While abstracting the feature detection, changes will need to be recorded in the environment over time. This is key to agents' collaboration in dynamic environments. It should be stated that it is not that this is impossible in multiagent SLAM, more that it is seldom considered.

2.3. The Voxel Grid and the Approach to Capture Data

Navigating unknown environments poses a significant challenge in robotics, given the complexity, dynamics, and diversity of real-world spaces [20,21]. To overcome this challenge, agents need a more universal view of the world to comprehend and represent the data they perceive, allowing for a common ground for information sharing and representation of surroundings in a simple and relatable manner. This approach has been explored in previous research, demonstrating the benefits of shared understanding of the environment for multiagent systems [13,22].

To enable agents to comprehend and represent the environment, various techniques have been developed, such as voxel-based representation learning [13,23] and semantic RGB-D SLAM [21]. Other studies have investigated deep loop closure detection and point cloud registration for LiDAR SLAM [24], fast and compact loop closure detection using 3D point clouds for indoor mobile mapping [25], and a lightweight LiDAR–inertial SLAM system with loop closing [26]. These techniques enable agents to represent the environment more accurately and efficiently, leading to better navigation and mapping performance.

Navigating unknown environments is a challenging problem in robotics, but various techniques have been developed to provide agents with a more universal view of the world, enabling them to comprehend and represent the data they perceive.

To reconcile the various approaches described above, this paper proposes an approach that aims to standardise environmental representations. To achieve this goal, it is necessary to adopt a standard form for representing complex environmental data, such as an occupancy grid. Occupancy grids offer a representation of the world as evenly spaced fields of probabilities. Initially set to zero, these probabilities are typically instantiated upon observation, allowing any unexplored location to be sampled and referenced relative to other grid locations.

In a traditional occupancy grid approach, the grid represents the probability of observing an object. To generalise this concept, the proposed approach transforms the variable z, a measurement observation, into a set of measurements $z = \text{set}$ and applies Bayesian inference to reason with this set. This approach levels the playing field for all heterogeneous agents, given that there is an agreed-upon set of descriptions (coordinates) to commonly address the environment. However, for our purposes, a 2D or 3D coordinate system does not sufficiently capture the complex nature of the data that needs to be represented. As an alternative, we propose using a 3D voxel grid, which contains the measurement and observation data inside each voxel. This voxel grid offers a rich representation of the environment, allowing agents to reason about the occupancy of different regions and the likelihood of objects existing within them.

This approach has been used in previous research, such as in voxel-based representation learning for place recognition [13] and semantic RGB-D SLAM for rescue robot navigation [21] These studies demonstrate the effectiveness of using voxel grids to represent complex environmental data and the benefits of doing so.

The world can be perceived as a set of probabilities that remain fixed, such as the likelihood of a region being occupied or containing a particular object. As time passes, this set of probabilities is updated, and information is stored together with a timestamp. Consequently, every voxel in the grid represents a collection of past observations regarding the environment. This approach enables agents to reason about different areas' occupancy and the likelihood of objects being present, even if the objects are in motion. By capturing data over time within a fixed space, the probability of an object's location becomes more certain as agents explore, while still retaining the data's validity despite dynamic and moving objects.

To convert from the image frame to the voxel, start with the agent's coordinate, which is assumed to be (0, 0, 0) if no prior information is available. Using the X, Y, Z coordinates of the pixel in the image and the pixel's depth (from the 3D depth sensor), the agent can project the pixel onto the world space grid, this can be seen in Figure 1. The information from the pixel will then be assigned to one of the neighbouring voxels. If there is already information present in the voxel for the current timestamp, the new and existing values will be combined and averaged. If the pixel is the first to be assigned to the voxel, all of its information will be added to the voxel for the current timestamp.

The processes that an agent undertakes in this paper are described in Algorithm 1. with details of these processes found in the constituent parts of Section 2.

Static definitions needed for agent loops
w = Camera Width
h = Camera Height

Algorithm 1: Agent loop

for *For each time step* T **do**
- Sync with Communication Server
- Record data from sensors
- Read odometer
- **for** *Generate point cloud. for each* X, Y *in image frame* **do**
  - depth data d
  - horizontal FOV f = horizontal FOV * (180.0 / π);
  - $\tan(x)$ = tan(horizontal FOV * 0.5);
  - Aspect Ratio $a = h$ / h;
  - vertical FOV $v = 2 * \arctan(\tan(f * 0.5) * AR)$;
  - Focal length $x_f = 0.5 * w * (1 / \tan(0.5 * f))$;
  - Focal length $y_f = 0.5 * h * (1 / \tan(0.5 * f))$;
  - Centre point for X $x_c = w$ / 2;
  - Centre point for Y $y_c = h$ / 2;
  - Run Image Segmentation Run Image Feature Detection Run Image Object Detection
- Translate based on odometer reading
- **for** *For each time step* X, Y *with Rotation* r **do**
  - $\begin{bmatrix} cosr & -sinr & 0 \\ sinr & cosr & 0 \\ 0 & 0 & 1 \end{bmatrix}$
- Local alignment of image or sensor fame at $T - 1$
- Convert results to voxel grid
- Global alignment of voxel grid position
- Alignment with Other agents
- Add voxel grid to world grid

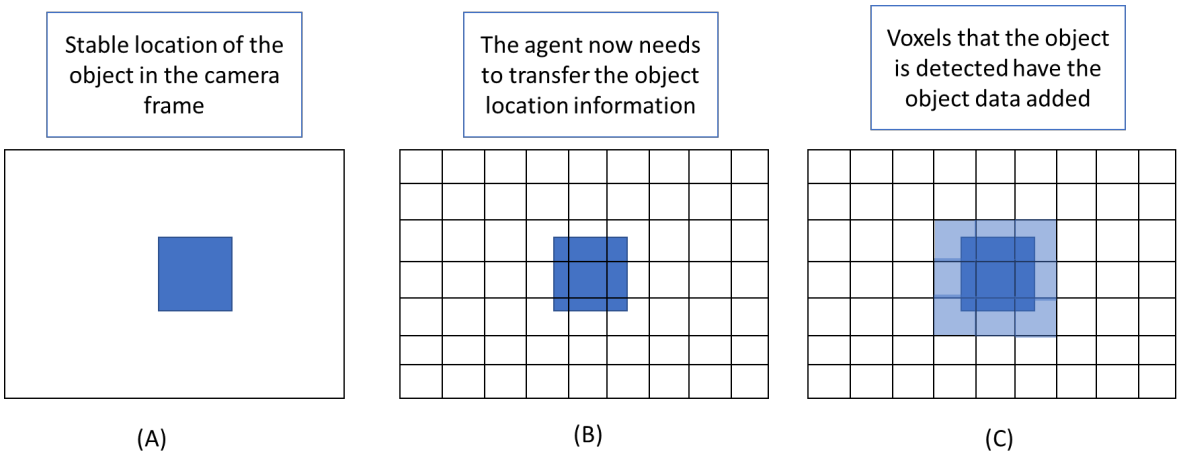

Figure 1. Figure illustrating an object overlaid with a voxel grid in 2D. (**A**) displays the object within its environment, while (**B**) depicts the voxel grid layered on top of the environment. (**C**) shows the voxels associated with the object displayed within the voxel grid.

2.4. Creating the Voxel Grid

The voxel grid is the key aspect to building the world up with vivid information, whilst still capturing growth as a probability over time.

If the agent has no prior understanding or mapping of the environment, then that agent's world can be started at any arbitrary point and defined as origin $(0, 0, 0)$. This is likely to change as the agent updates its information and communications with other agents. If it is now assumed that a data file was obtained with the current frame of vision from the

agent described in X, Y, Z Cartesian space, the origin point in the data file now becomes the origin point in the world view of that agent, and, furthermore, the populated voxel grid world now will use that point as the starting point to fill the world with unexplored voxel probabilities. Where the world space origin is not important (any point in a space can be defined as the originally defined point), it is clear that what is most significant is the constituency of the data aligned to that point, this being achieved with the aforementioned alignment techniques.

Now that the origin is defined, the voxel generation process should be considered and the information contained within. First, the most basic information—positioning—can be inspected. After the alignment of one frame to another, the agent can simply take the point in 3D space and update the world space. However, this method would be too fine in resolution and we would rather combine information from close neighbours to generate a more generalised view of the information. For this, a region of interest can be simply defined and information can be captured from any point inside that ROI. This is not dissimilar to downsizing, but, rather than an average of the points' locations, it is the mode of where more points fall in the world space voxel grid.

Using the above methods, the agent has a system in which to recreate an SLAM system using frame-to-frame alignment. However, as stated at the start of this paper, the world is already a fixed grid of voxels; understanding the relationship between the perceived sensor data and that voxel grid links these data points together. The agent must now update that world space coordinate with the information it can gain.

One might look at this approach and ponder on the differences between the SLAM system, or, indeed, other mapping systems that align 3D point clouds of data. The response is that with the voxel grid system, the world state does not move with new alignments; that is to say that when an agent revisits an area and the coordinate is perceived as different, the map the agent has generated is not moved or aligned, but, rather, the information is fed into the voxel at the new coordinate. An example of this is shown in Figure 2

First, a more transitional way of aligning and registering mapping data will be examined. For simplicity, the example will be limited to 2D.

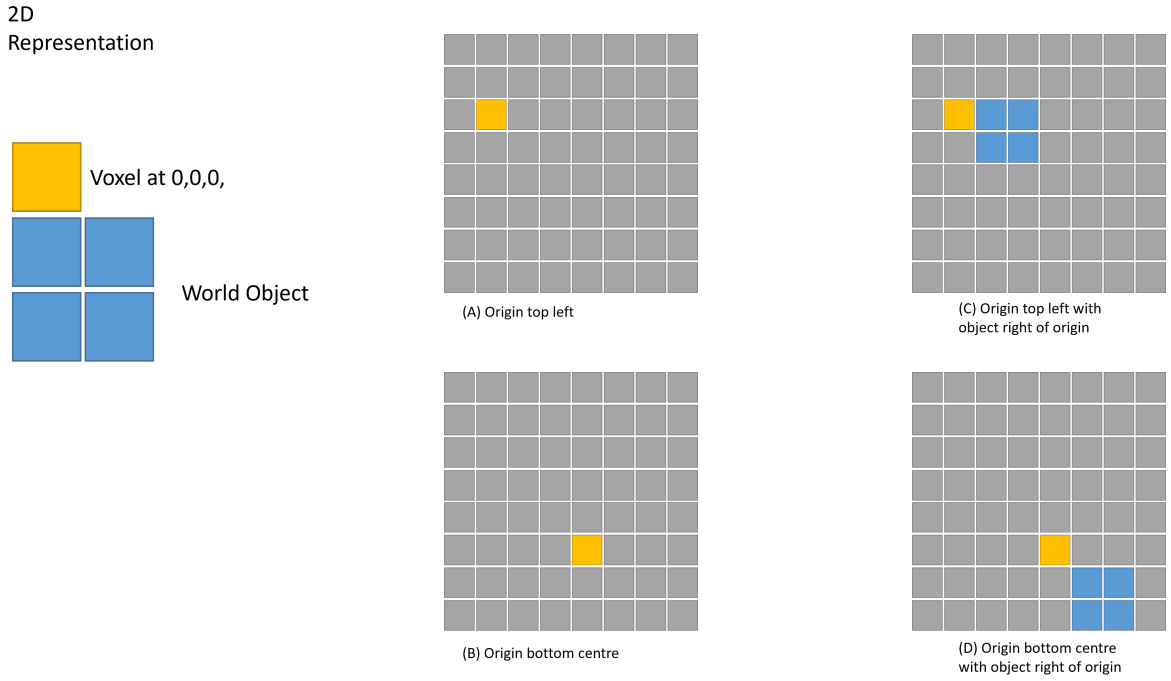

Figure 2. The figure shows the origin can be set arbitrarily when the agent first starts recording a new environment. .

As can be seen in the above Figure 3, as the data from the agent become misaligned, the data are reconciled to the new location relative to the first frame. This, in effect, relocalises the environment to the new frame, and the information about the "incorrect" data is lost.

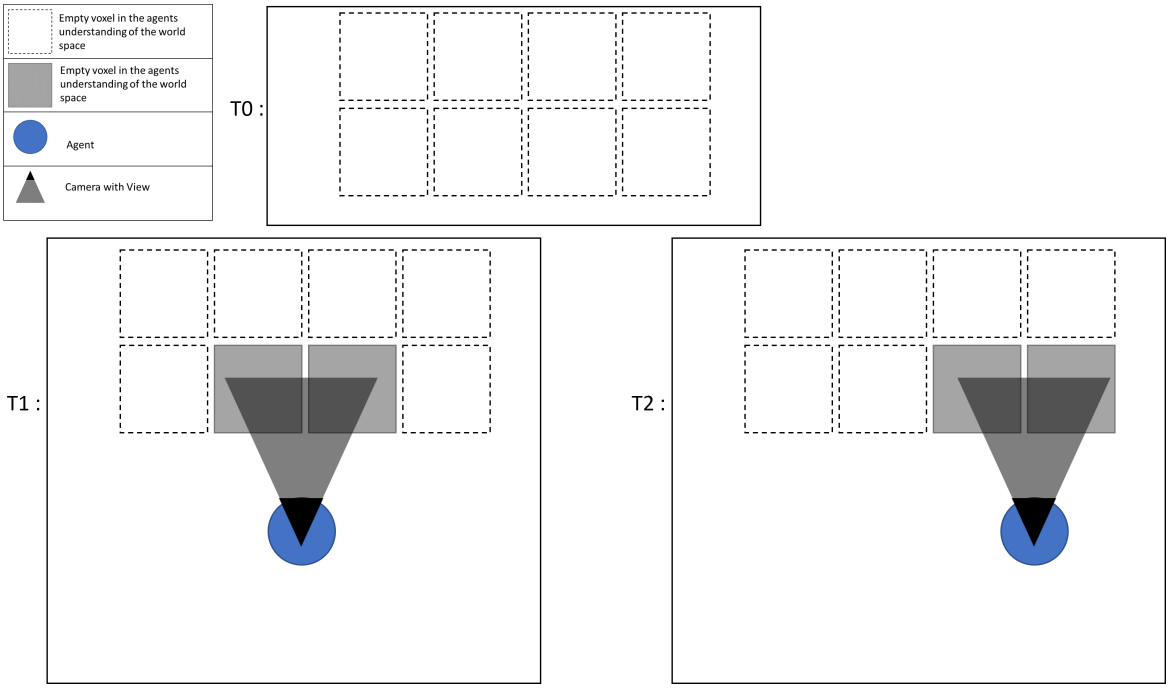

Figure 3. Figure illustrating an agent initializing two different sets of voxels based on the position of the environment.

In Figure 4, we can see that the data are still being realigned; however, the information about the previous records is kept, and over time, this will provide a greater probability for a given voxel to contain the ground truth information.

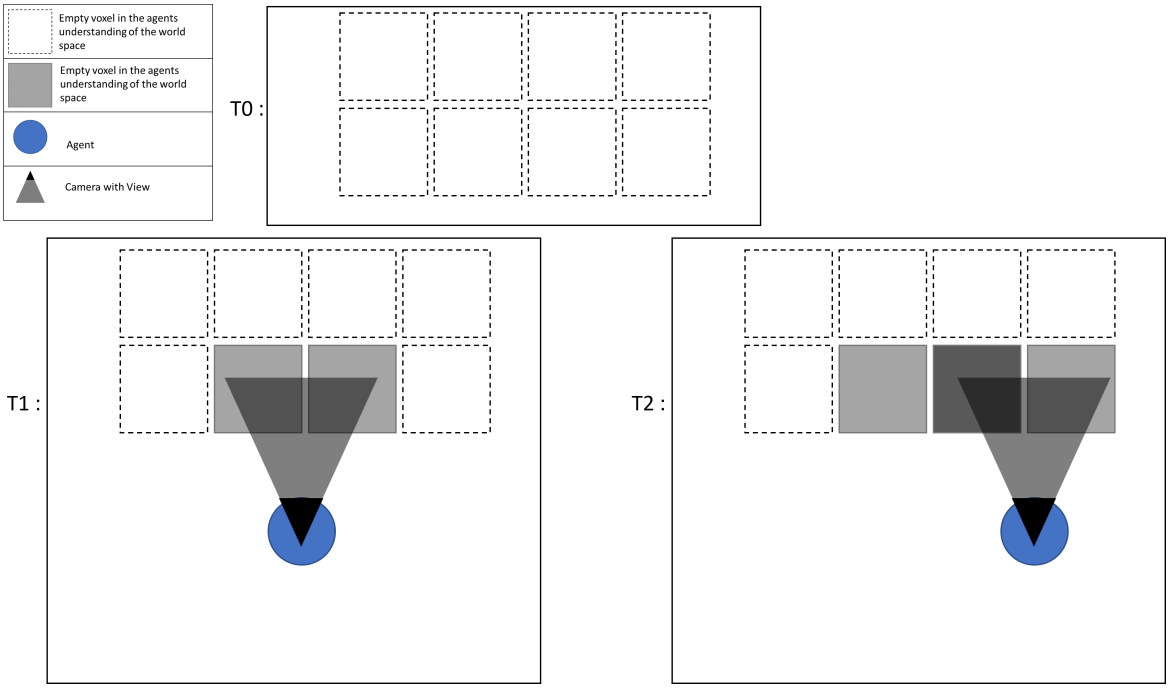

Figure 4. Figure showing object overlay increasing the probability of a voxel containing an object.

The voxel— A voxel is a unit of graphic information that defines a point in three-dimensional space. For use in this paper, the information contained inside is extended from just a simple X,Y, Z, and colour, to the storage of information that will help define the world's complex nature. The agent should store information pertaining to its observation of the environment, for example, heat or sound data. Figure 5 shows this in more detail. In this paper, some assumptions are made about the data that are recorded and stored inside the voxel; however, there is no necessity to limit the recorded information. The below shows the definition of the voxel grid utilized in this paper.

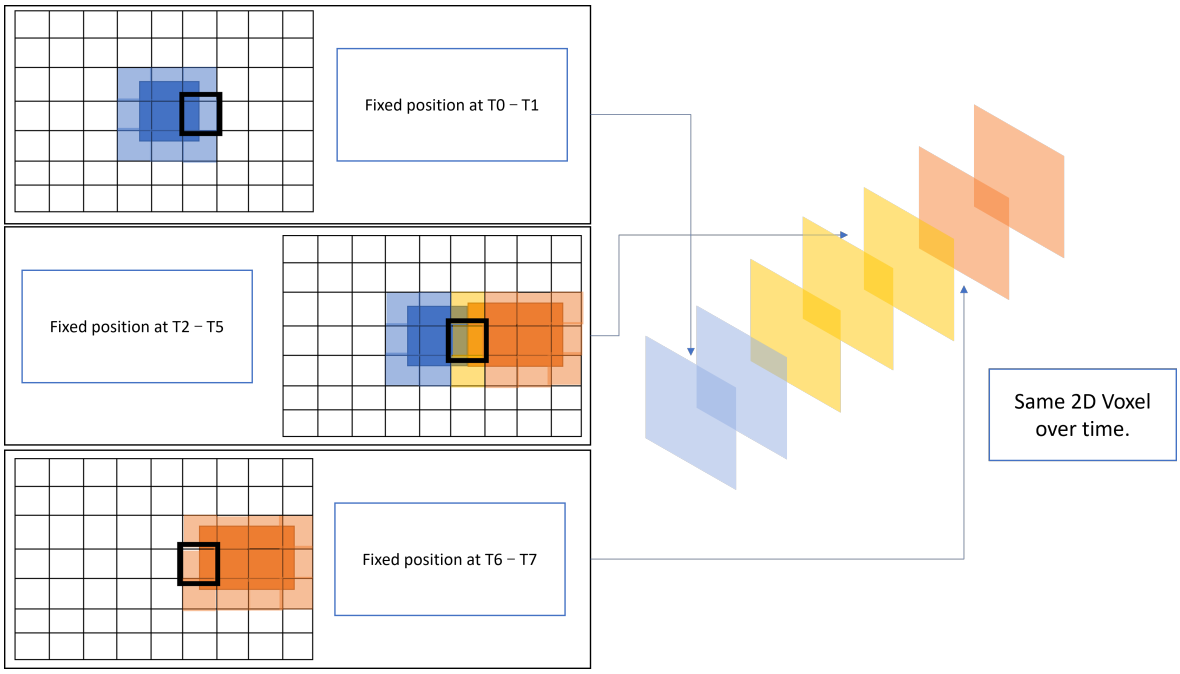

Figure 5. Figure showing how data of a single voxel are an array based on time.

w = The size of the world
Time t is an arbitrary, but consistent, timestep that is defined by the agent.
g = { $v_1, \ldots, v_w$ }
v = { x,y,z,d }
where { X, Y, Z} $\in \mathbb{R}$
and represents the position of an arbitrary point in Cartesian space.
d = { $h_1 \ldots h_t$}
d = { e, r, b, j }
j = { $o_1 \ldots o_s$ } Where s is the number of objects discovered in any given frame
o = { i, p }
p = The probability of a given object being in that voxel at time t
e = Agent Experience
r = Reliability of agents
b = Agent Bias

Object detection from a recognised set—A key aspect identified in the hypothesis is that of object recognition under occlusion. That is, it is difficult to identify objects that are partially obscured, by either other items within the environment or perhaps poor viewing angles. The voxel grid can address these issues by allowing the object being identified to be reconciled with a template for the given object to be found, over multiple orientations and scales. The voxel grid also provides wider net data points, as, rather than captured visual data being points x,y,z in a space with no determined size, they fall into a discrete grid of voxels with a defined size. This in turn means that when an agent is scanning for an object it has an increased probability of catching a feature, an example of this is shown in Figure 6

with the real data comparison shown in Figure 7. It should be noted, however, that this comes at the cost of the resolution of the visual data.

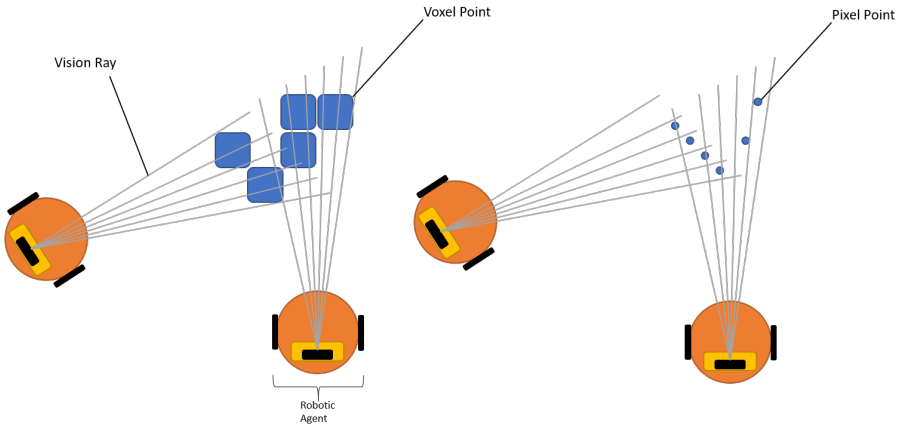

Figure 6. Figure showing the voxel approach to collecting data points.

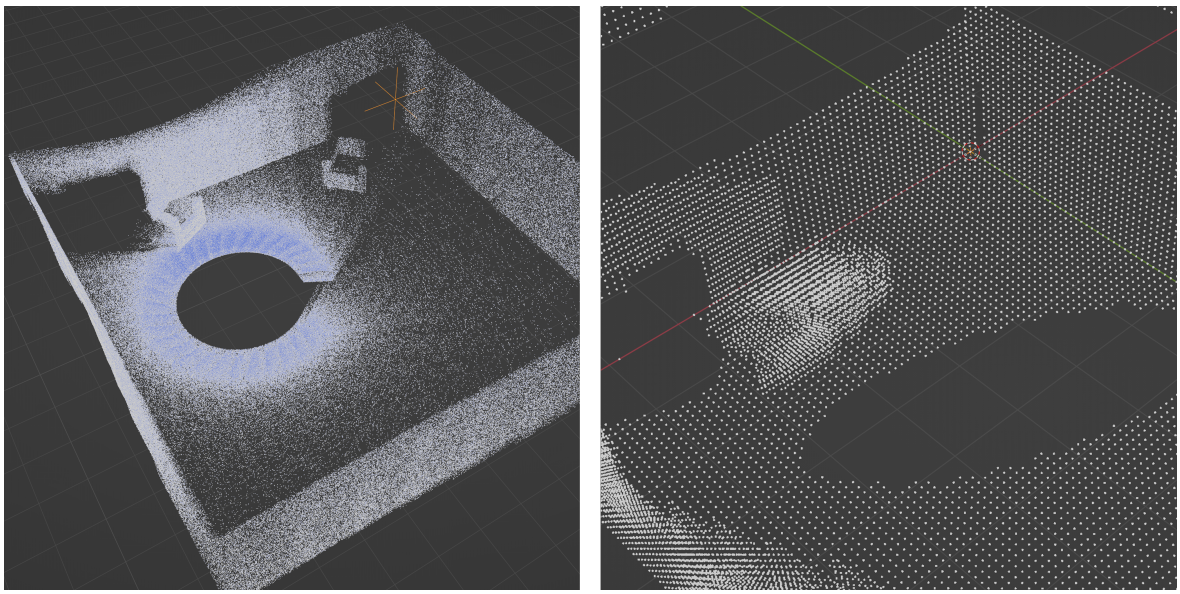

Figure 7. Figure showing point cloud and voxel grid.

2.5. Loop Closure and Object Probability over Time

The closed-loop problem that is often present in SLAM is not realised in the same way as traditional methods. As the voxel grid method is already a fixed set of states, or voxels, then there is as such no loop to close; however, the same issue, in principle, can be seen, whereby the voxels that contain object information will drift in a direction overtime and voxels will not link together to create complete environments. In fact, it could be argued that this drift in object detection would be significantly more of an issue as the objects would appear to be dynamic in nature, due to the recording of drift in the voxel information over time.

In the example of a car driving a set distance, then the problem can be explained in more simple terms. In this example, the car has instructions to drive 10 metres, then turn 180 degrees over another 20 metres, and, finally, repeat. This creates a "loop" whereby the car returns to the position it started in, this is shown in Figure 8. In the first image, this can be seen in effect. The image shows the car with perfect odometry and therefore returns to the same starting position. However, in the second image, we can see that the car has an error on the internal distance and rotation reading and, therefore, even though the car believes it is back to the same position it started at, the ground truth shows a different story and, in fact, the car has not returned to the starting position, leaving an "open" loop circuit.

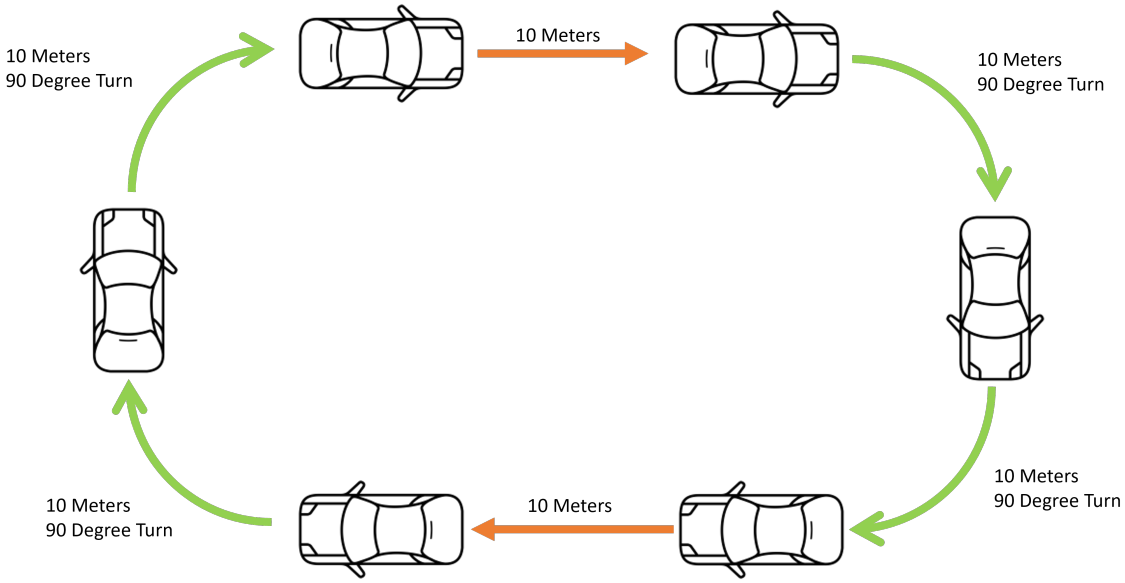

Figure 8. Figure showing the perfect closed-loop system.

Below, it can be seen what happens when a small error is applied to the reading. The error propagates throughout the processes and the misalignment becomes significantly worse over time.

With the voxel grid approach described in this paper, there is not a closed-loop detection, as such. When the agent returns to a position it could consider repeated or closed, the mapping data are not aligned, shown in Figure 9.

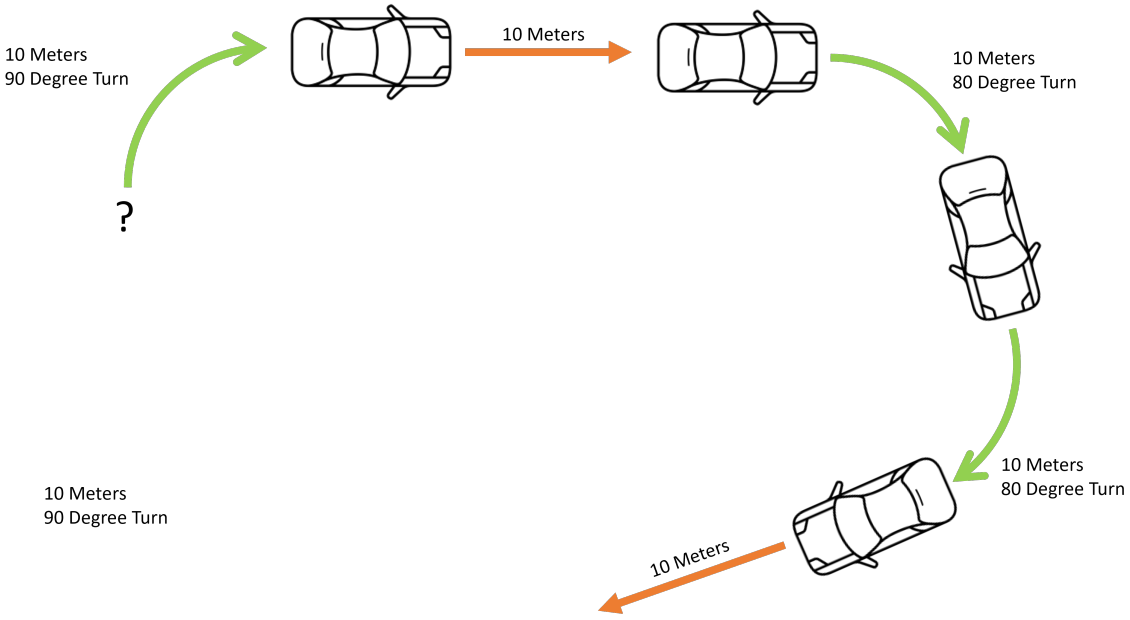

Figure 9. Figure showing the imperfect closed-loop system.

2.6. Centre Point of Object, Clustering

Object centroid—To utilize the known objects in the world for aligning and scaling the voxel grid, a fixed center point is required for each object. This center point serves as a reference point for applying transformations, assuming that we have at least two identifiable objects. The process of generating identifiable objects is an area of ongoing development and will be explored in future work for more effective methodologies. Figure 10 shows an image of the concept. Figures 11 and 12 shows a real data and graphical plot respectively.

Clustering—To determine the center point, the object data is clustered based on the voxels containing each object. This helps to identify the center of the voxel cluster belonging to each object. We use a simple K-means clustering algorithm, with K equal to the number of objects present in the environment. Any voxels identified as outliers are disregarded during the clustering process.

The below defines the clustering of objects within the agent's voxel grid.

$$\arg\min_{S} \sum_{i=1}^{k} \sum_{x \in s_i} d(x, \mu_i) \tag{1}$$

where
S is a set of voxels within the voxel grid;
k is the number of unique objects within the environment the agents has mapped;
x is an observation data point;
μ_i is the mean of points in S_i
$d(x, \mu_i)$ is some arbitrary function that measures the distance between points in any given unit.

In this case, the Euclidean distance function will be used, given by

$$\arg\min_{S} \sum_{i=1}^{k} \sum_{x \in s_i} |x - \mu_i|^2 \tag{2}$$

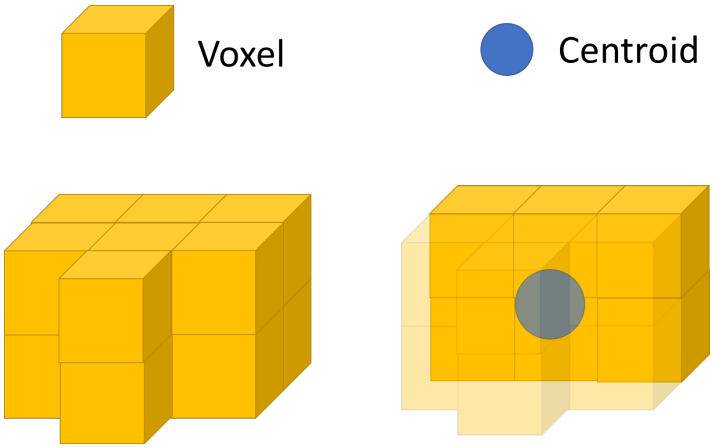

Figure 10. Figure showing centre point of a detected object.

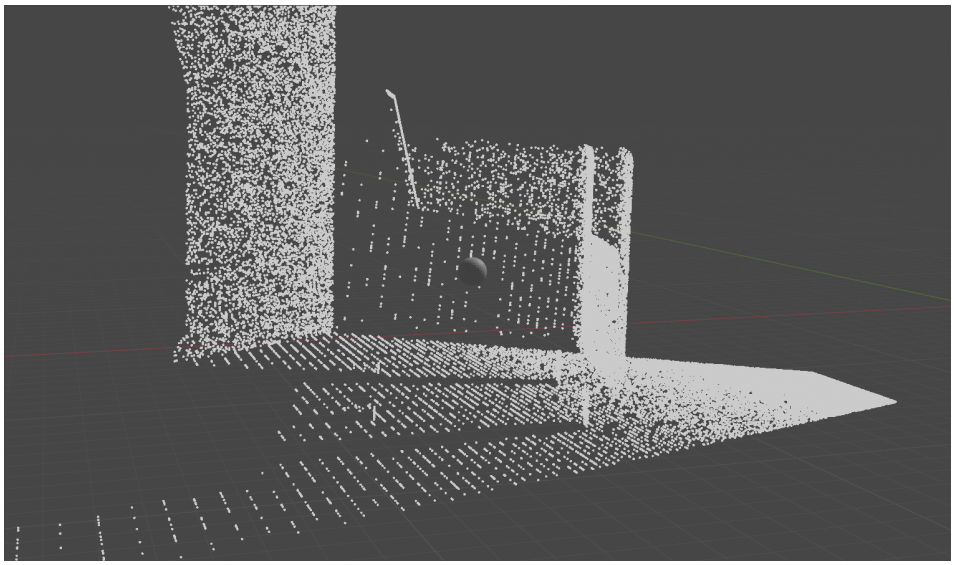

Figure 11. Figure showing centre point of a detected object with real data.

The below images show data taken from a single frame with all voxels removed that did not contain an object. The data show a frame containing two objects.

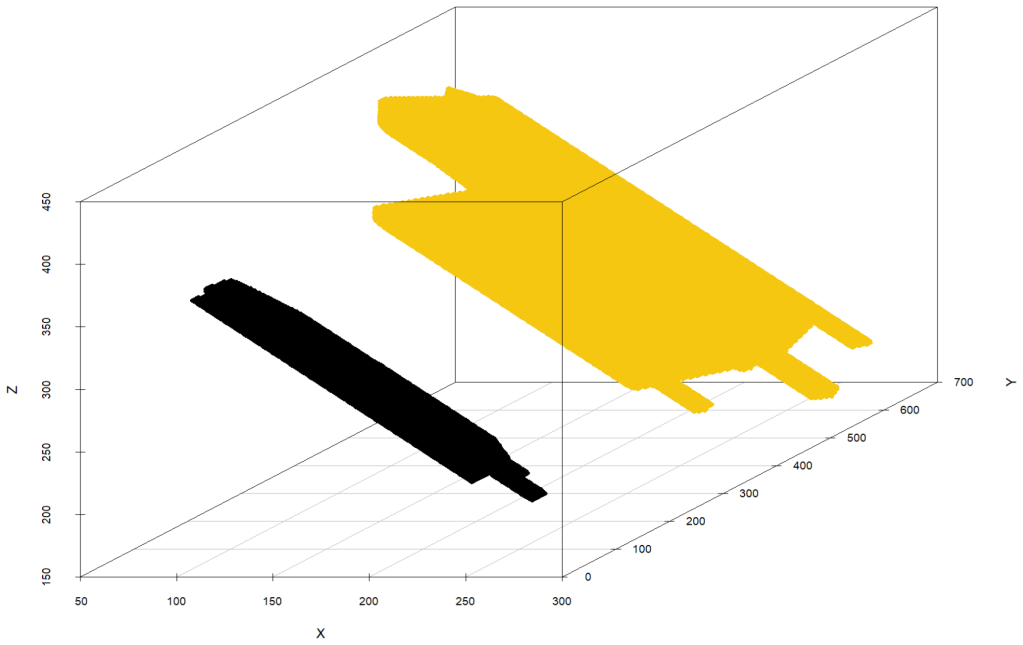

Figure 12. Plot of K-means clustered data of two separate objects.

2.7. The Frame-to-Frame Alignment

To align from two frames, the agent considers two key aspects of information. The first is the internal odometer and the second is the holographic difference between the two frames. The odometer would, in principle, be the only information required, as by considering how much the agent has moved in the x, y, and z plane, while also considering the rotation, an updated position of the agent could be determined in the global world space. However, outside of the simulation, there is no perfect odometer information and an error will also occur. This error will have a cumulative effort on the perceived position of the agent, and unless the error is near-perfect Gaussian, it will lead to an open-loop mapping. In SLAM, this problem is solved using an extended Kalman filter, and in the proposed approach, that is also what is being utilised, where the frame-to-frame alignment is a prediction based on the odometric reading and the discrepancy of features detected in the frame.

$$O = [X, Y, Z, Rotation, Scale] \tag{3}$$

Each time the agent moves, the odometer of the agent is merged with the frame alignment process. Note that this fusion could be complemented with other forms of localisation such as GPS. This will be further covered in the section relating to heterogeneous agent design.

Let us break down the frame alignment technique into steps and walk through the processes. Figure 13 defines the scenario for the subsequent explanation of frame-to-frame alignment.

Step 1—is to find all image features; these are our key points to connect between one frame and another. $f_t = x \mid x$ SURF Feature. This can be seen in Figure 14.

Step 2—The agent moves; this movement can be in any form of translation or rotation, but for this example, the movement will be limited to a linear transform in the X plane. This can be seen in Figure 15.

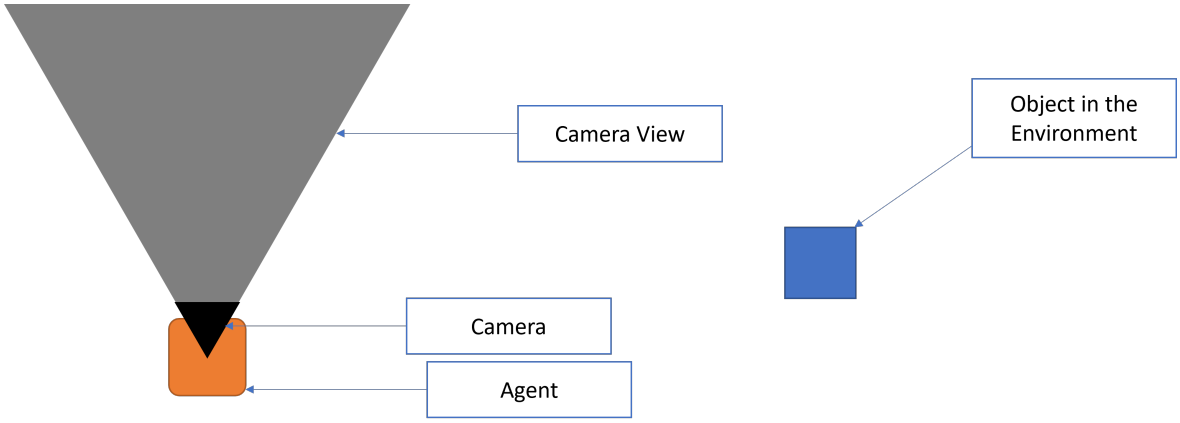

Figure 13. Figure showing key for agent alignment images.

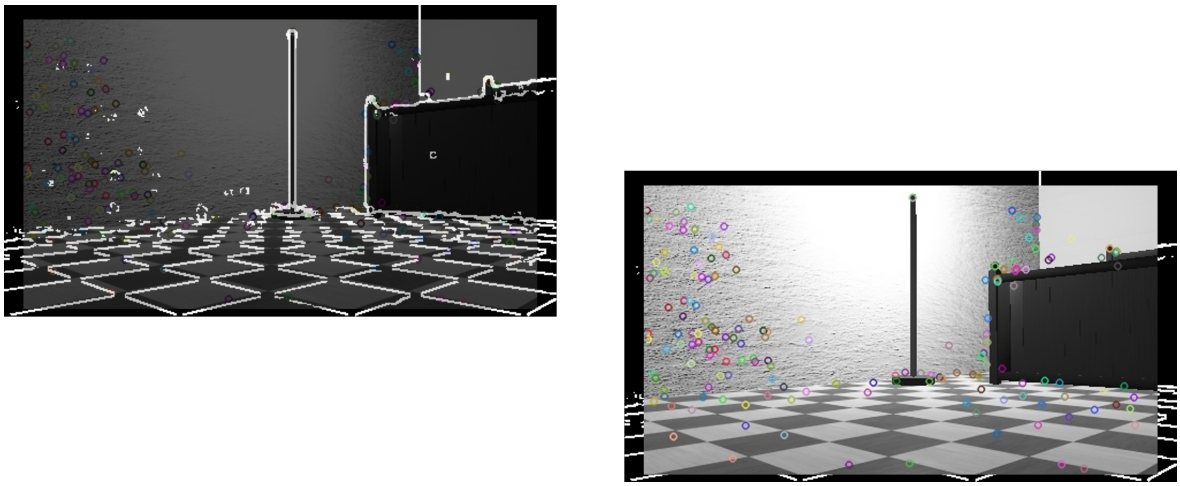

Figure 14. Figure showing features detected in frame.

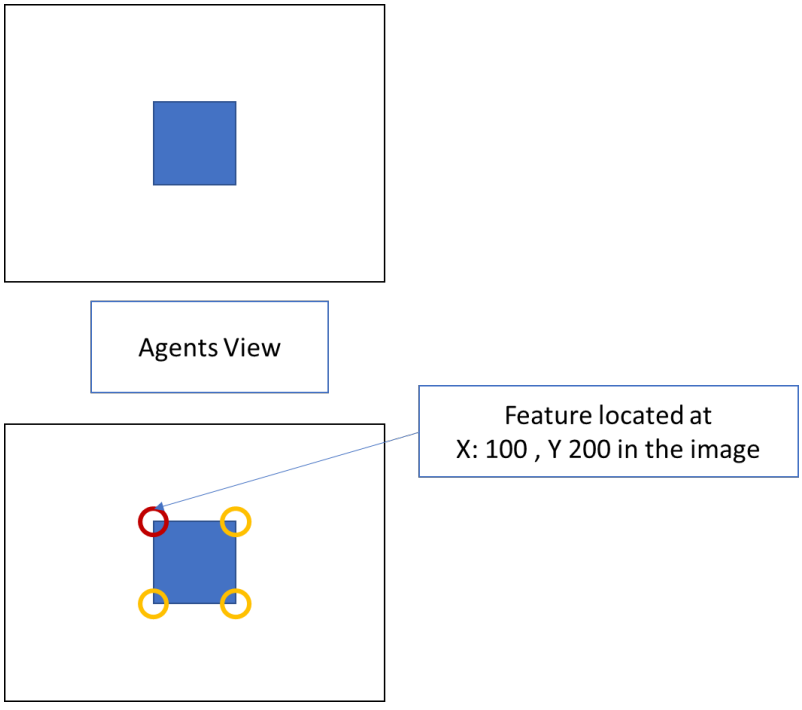

Figure 15. Figure showing features detected from agent's view.

Step 3—Compare the difference between the original image frame and the agent's new location image frame with respect to the matched features detected. $h : f_{t-1} - f_t$. This can be seen in Figure 16.

Step 4—Compare the agent's odometric translation with respect to the matched features. $o : f_{t-1} + CurrentOdometerreading$. This can be seen in Figure 17 and further explained in Figure 18.

Step 5—Take a final weighted value from the two comparison figures. This can be seen in Figure 19 with the full algrothem broken down in Algorithm 2.

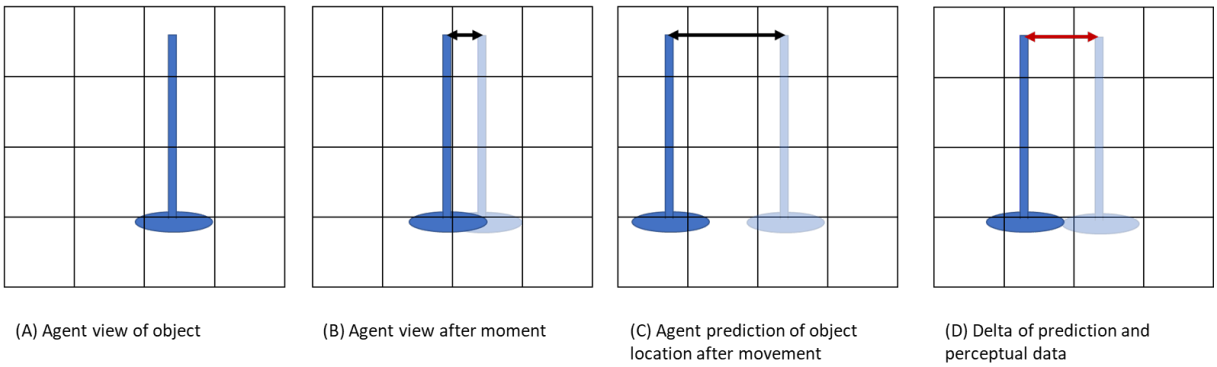

Figure 16. Figure showing the adjustment as the agent moves.

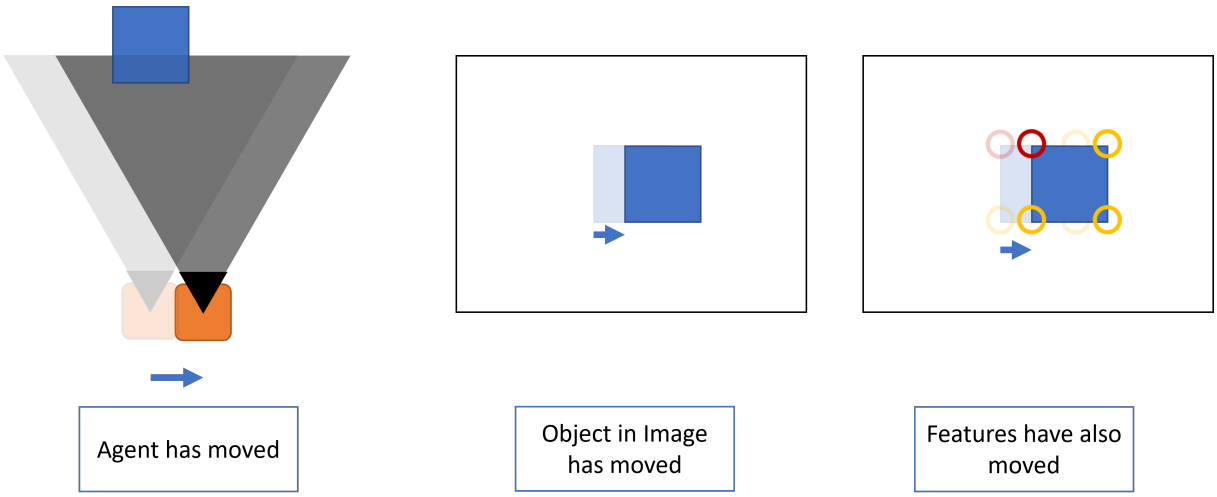

Figure 17. Figure showing the agent has moved and the view of the object has been updated.

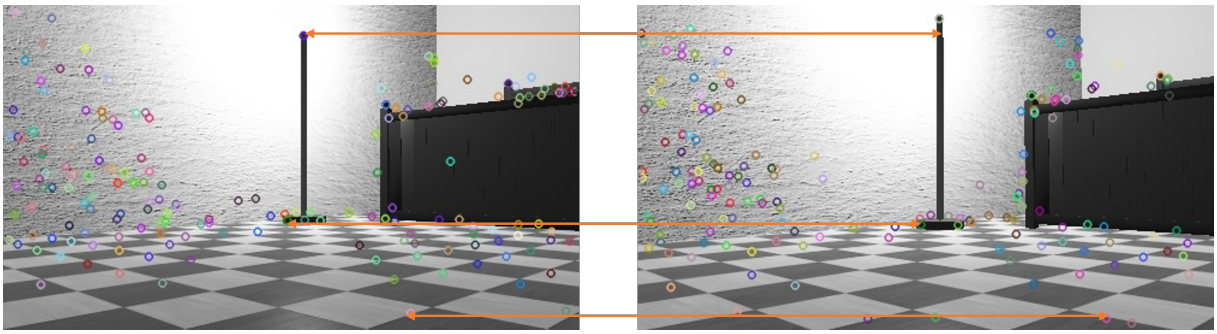

Figure 18. Figure showing the feature matching to determine local adjustments.

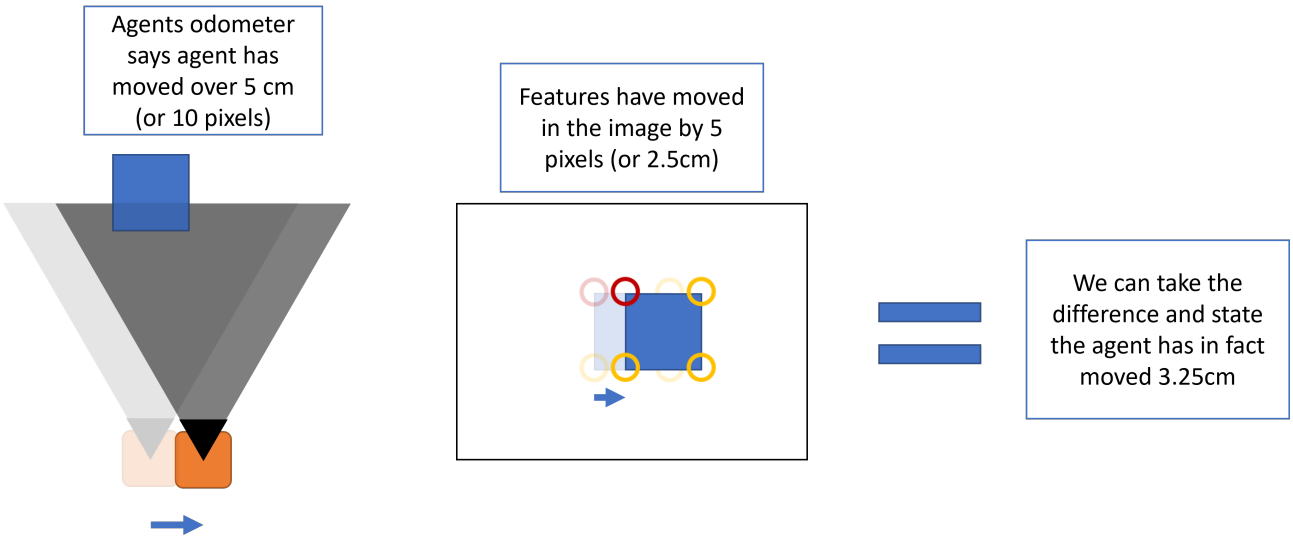

Figure 19. Figure showing measurement of differences in where object features are in the world vs. the prediction based on the agent's odometer.

From here, the agent has defined the image frame in relation to where it is in the environment. This is still a local reading to the agent and not a global understanding of where the frame is until such time that the data from the image frame are placed into voxel form and on the environment voxel grid. These are the next setup steps to be considered.

Algorithm 2: Frame-to-frame alignment

Locate the image at t −1
Gather the current image at t
Find features in images
1 **for** *For each image pixel (x,y) in Image_t and image_t* −1 **do**
  Detect key points in image or LiDAR reading
  Calculate the movement of key points based on odometer reading
  Read the difference between key points from predicted to actual
  Translate agent position based on updated prediction
Match local features and measure differences
2 **for** *For each set of features (x,y) in Image_t and image_t* −1 **do**
  Detect key points in image or LiDAR reading
  Calculate the movement of key points based on odometer reading
  Read the difference between key points from predicted to actual
  Translate agent position based on updated prediction

2.8. Any Local Object to Global Map Alignment

Global map alignment—The global alignment step is where we closer align to the principles of SLAM. The frame-to-frame alignment should offset any issues with combining volumetric mapping data; however, errors still need to be addressed in the agent's position relative to the world. For this alignment, landmarks in the voxel grid are used and measurements are taken at any given timestamp to locate and adjust the agents' position in the world. This process of global map alignment is broken down in Algorithm 3.

Algorithm 3: Global alignment

Cluster objects in voxel grid at time T
Remove outliers from object data
Alingment
for *For each time step T* **do**
    Cluster image voxel gets for different objects Define centroid for objects
    Measure the Euclidean distance from the agent position to the centroid
    Measure the Euclidean distance from the agent perspective to any visible centroids Communicate with other local agents if that agent can view the position of the current agent. **for** *For each agent A* **do**
        Cluster image voxel gets for different objects Define centroid for objects
        Measure the Euclidean distance from the agent position to the centroids available Measure the Euclidean distance from the agent's *A*'s of all objects and the original agent Calculate difference Adjust to minimise the error
    Calculate difference Adjust to minimise the error

2.9. Multiagent Alignment of Maps

Multiagent environment mapping—Multiagent environment mapping is a critical area of research in robotics that focuses on utilising multiple robotic agents to collaboratively map, assess, and interact with dynamic and complex environments. This approach is particularly valuable in situations where human intervention is risky or impractical, such as during disaster response or exploration missions. Key aspects of multiagent mapping include effective communication among agents, decentralised data fusion, and consensus algorithms. A good example can be found at [27]. The overarching goal is to create a comprehensive and accurate representation of an environment, leveraging the strengths of each agent while overcoming individual limitations.

The centralised communication method, utilising a server or broker, offers several benefits, including improved coordination among agents, simplified data package management, and a clear focus on specific goals for the agents.

However, these advantages come with trade-offs in three primary areas:

- Agents must be predefined and registered with the central coordinator before joining the colony or domain.
- Errors occurring within the system may propagate throughout the entire colony.
- As the number of agents increases, the centralised system must manage vast amounts of data to maintain coordination among the expanding agent population.

Contemporary efforts by [28] propose a hash-graph-based framework for common knowledge. Although there is a recognised need for a common knowledge base in the research space, no consensus has been reached on the ideal solution. Referring back to [28], an early consideration was the use of blockchain for a common knowledge base. One identified weakness of this idea revolved around the "majority attack", in which a single entity controls 50 percent of the network and, consequently, dictates the entire blockchain. This issue undermines a crucial aspect of transitioning to decentralised communication. While the proposed approach in this paper addresses this problem, the transition-based approach overlooks the potential to examine all the data in the environment. By only recording data deemed useful or necessary to share, it misses opportunities to identify trends in areas where no change has occurred. In contrast, the voxel grid approach presented in this paper offers a more generalised solution, enabling agents to observe trends even in areas with no observable changes. This is achieved by storing historical information inside each voxel that builds over time.

The Algorithm 4 shows the communication approach used in this paper.

The Algorithm 5 shows how the agents combine their mapping data.

Algorithm 4: Message sync

for *For each time step T* **do**
- Sync with Communication Server
- Check for pending messages about the environment in the Environment Buffer
- The communication server registers the agent environment and simulates close agents for communication
- The communication servers check for messages from other agents in the buffer
- The communication server passes other agents' messages to the asking agent
- Mapping data is compared and aligned by asking agent
- Changes to voxel grid sent back to communication server and held in environment buffer;

Algorithm 5: Combine agents' maps

Agent sends mapping data for the local environment to another agent
if *at least 2 common object exists* **then**
- minimise the distance between all object centers **for** *For each agent A* **do**
  - Cluster image voxel gets for different objects Define centroid for objects
  - Measure the Euclidean distance from the agent position to the centroids available
  - Measure the Euclidean distance from the agent's *A*'s of all objects and the original agent
  - Calculate difference
  - Adjust to minimise the error

Object centre point—Now that there is a defined centre point for objects, or landmarks, within the environment data, the agent can utilise this to align the agent's pose and location, this is shown in Figure 20 The agents should also be able to use the mapped-out understanding of the environment in order to localise. This is much more in the order of traditional SLAM; however, with further work in this topic, there is an intent to use an ontology overlaid onto the topology, and this could play a significant part in the detection of features within the image.

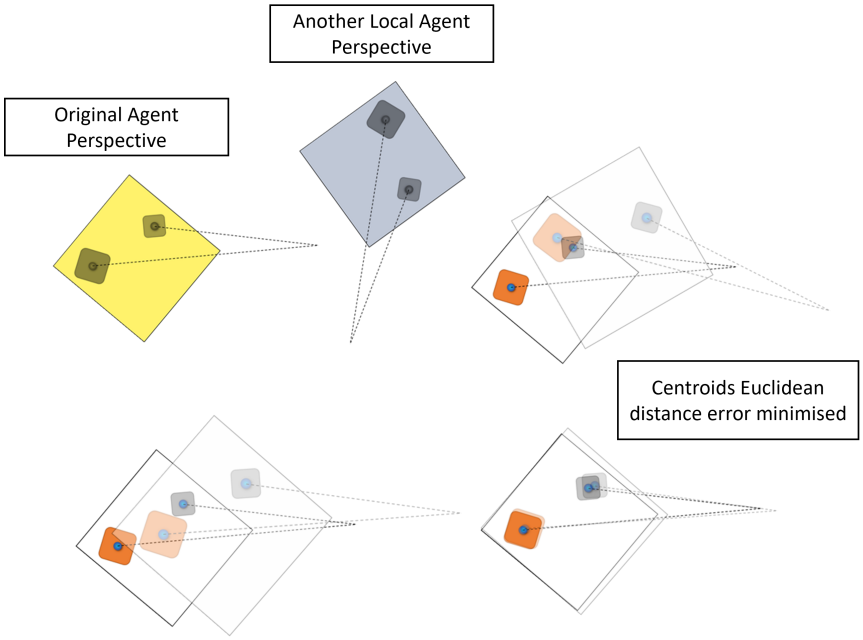

Figure 20. Figure objects being used for global alignment of the agent.

This paper provides an example of aligning LiDAR and 3D data, shown in Figure 21, demonstrating how this can be achieved. However, it is worth noting that the paper assumes that objects are successfully detected when gating perceptual data. Although this assumption may raise some concerns, it should be considered in the context of the numerous object detection methods available for both LiDAR and 3D depth data. Nevertheless, further research is necessary to explore this topic in more detail and improve the accuracy and reliability of object detection in complex and cluttered environments.

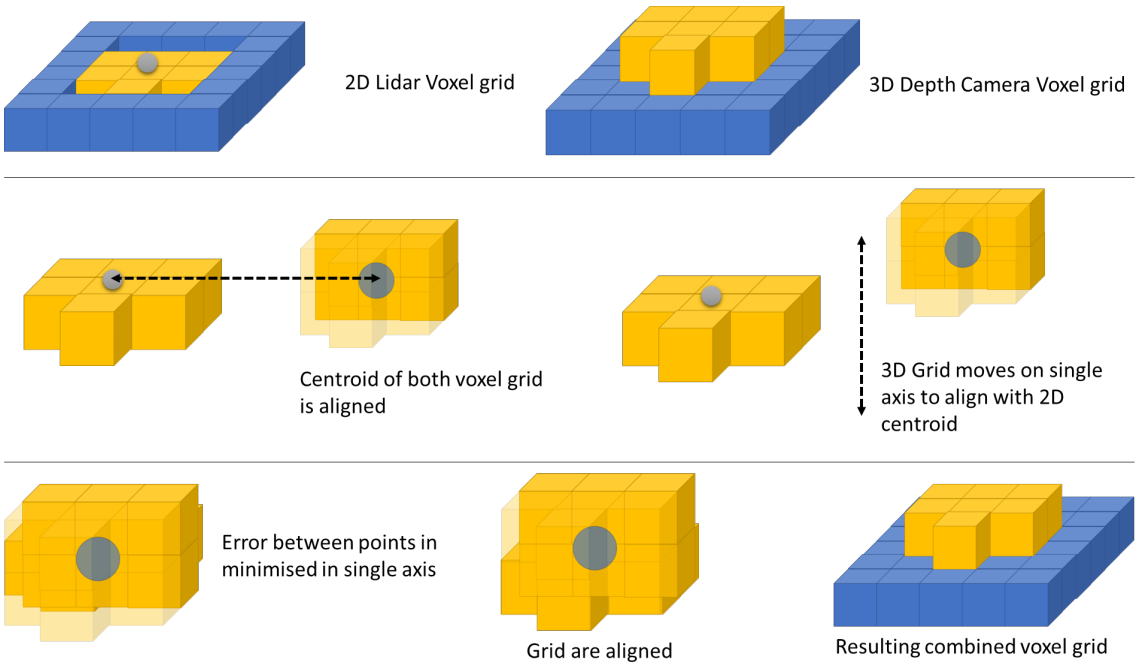

Figure 21. Figure for alignment between 2D and 3D agents.

3. Results and Discussion

The results written below are simulated in the WebBotes virtual environment. This software is a physically based simulation environment that can be used to run scenarios from a number of robotic agents in custom-built environments. In this instance, a Pioneer 3D-X was replicated.

The Pioneer 3D-X is a mobile robot platform designed for autonomous navigation and mapping tasks in indoor environments. It is equipped with a range of sensors, including a 2D laser scanner, a stereo vision camera system, and an inertial measurement unit, which enable it to accurately perceive its surroundings and generate a 3D map of the environment. The robot is powered by a high-performance computing system that enables it to perform real-time data processing and decision making, and it is capable of navigating through complex environments with obstacles and dynamic changes.

The following results are categorised into three sets, each measuring the pose estimation of a single agent. An initial image is provided below to establish the ground truth of the measured agent and to provide a measurable baseline. All agents employ the approach outlined in this paper. The first set of results shows a single agent using a 3D depth camera. The second set presents the results of a single agent using LiDAR. Finally, the third set of results showcases the same LiDAR agent, but this time utilising the 3D data from the other agent.

Figure 22 shows the ground truth of the measured agent.

Figures 23 and 24 are showing the single 3D agent.

Based on Figure 25, it is evident that the error in the pose estimate escalated rapidly and almost exponentially. Although this error curve may decrease slightly over time, it does not align with our expectations. This single agent is operating in a simulated and relatively sparse environment, which makes it challenging to localise accurately.

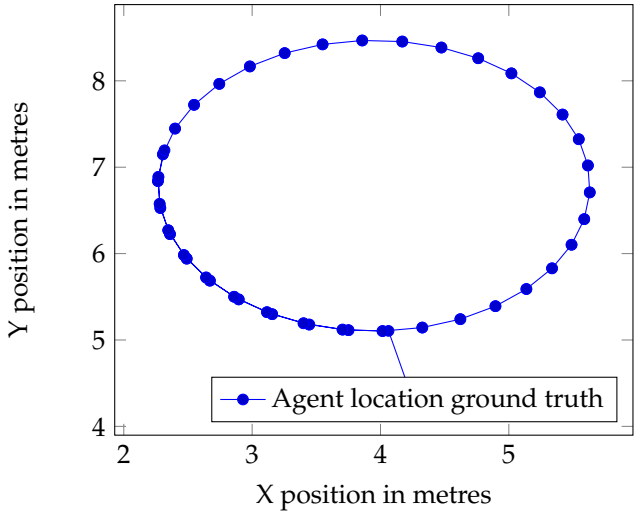

Figure 22. Data of ground truth of the measured agent

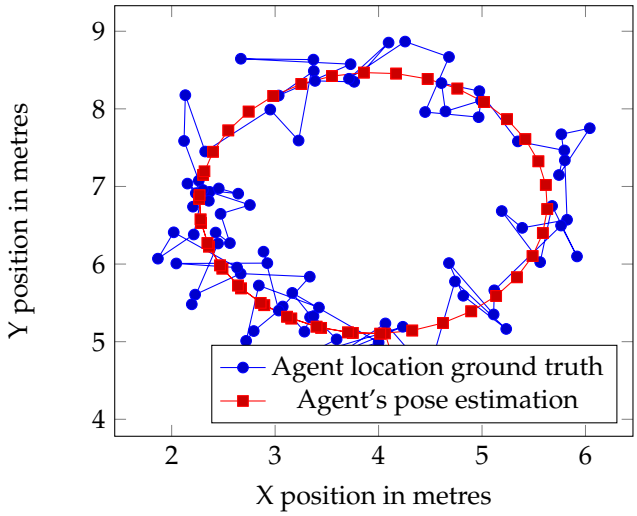

Figure 23. Data of 3D agents position estimation.

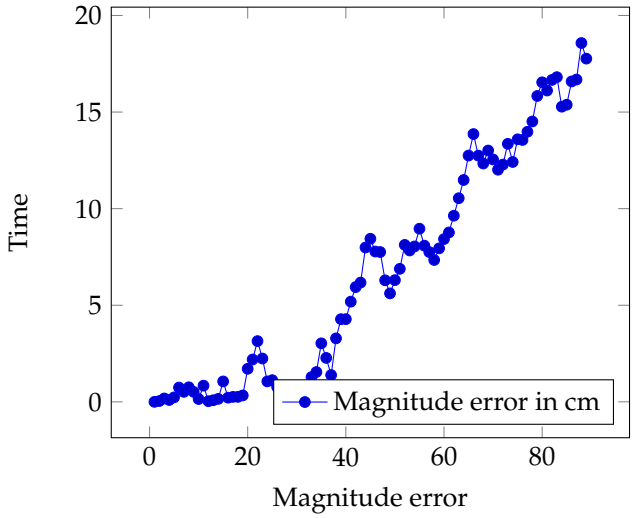

Figure 24. Data showing error over time for 3D Agent.

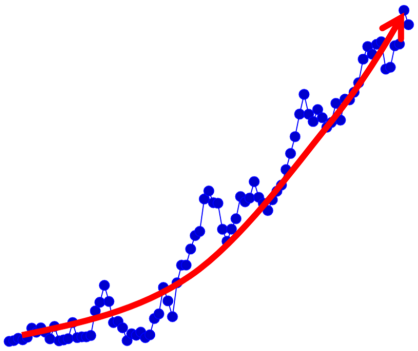

Figure 25. Figure with arrow indicating the high growth rate of the error in a single agent pose estimation.

Firgures 26 and 27 show a single LiDAR agent.

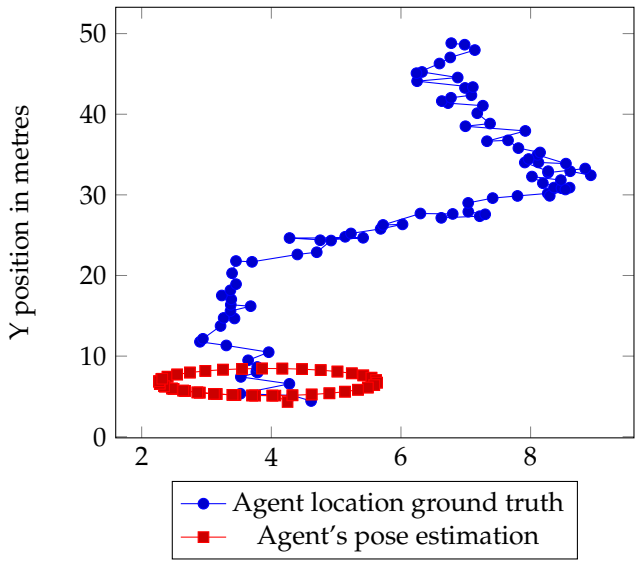

Figure 26. Data of 2D agents position estimation.

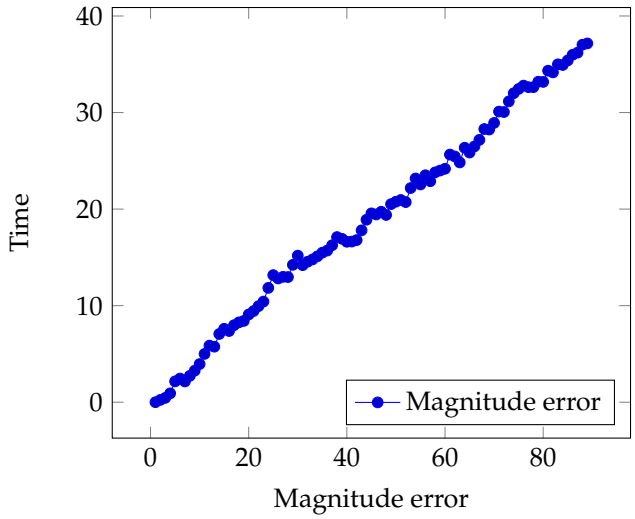

Figure 27. Data showing error over time for 2D agent.

Figure 28 and 29 is the final set with the combined effort.

The results shown in Figure 30 is a comparison of the three result sets, demonstrating how the multiagent voxel map generated improved results compared to single agents.

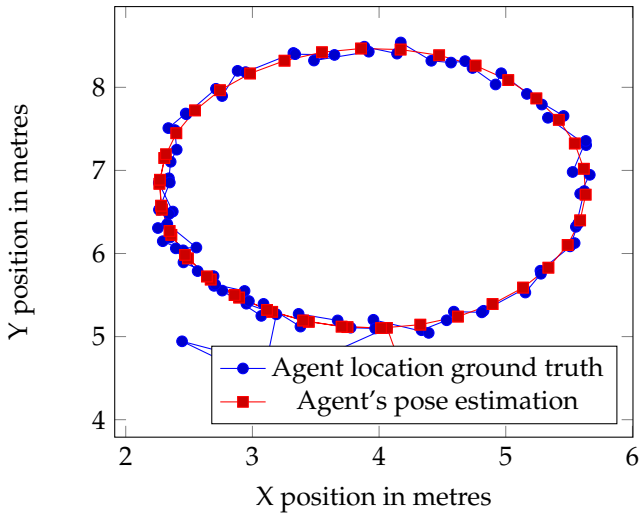

Figure 28. Data showing combined effort of the agents.

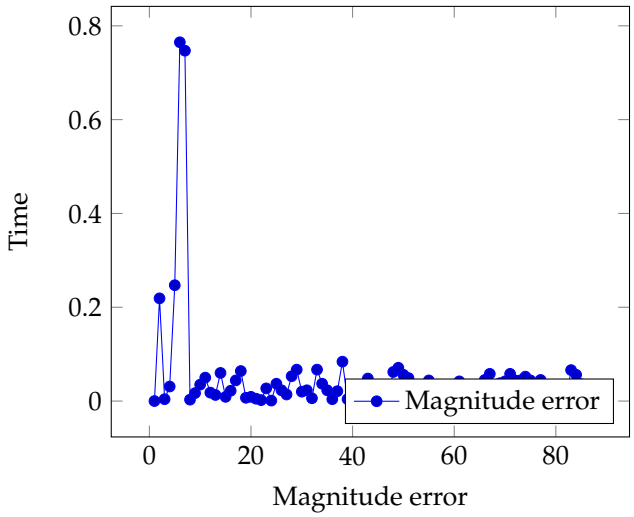

Figure 29. Data showing error over time.

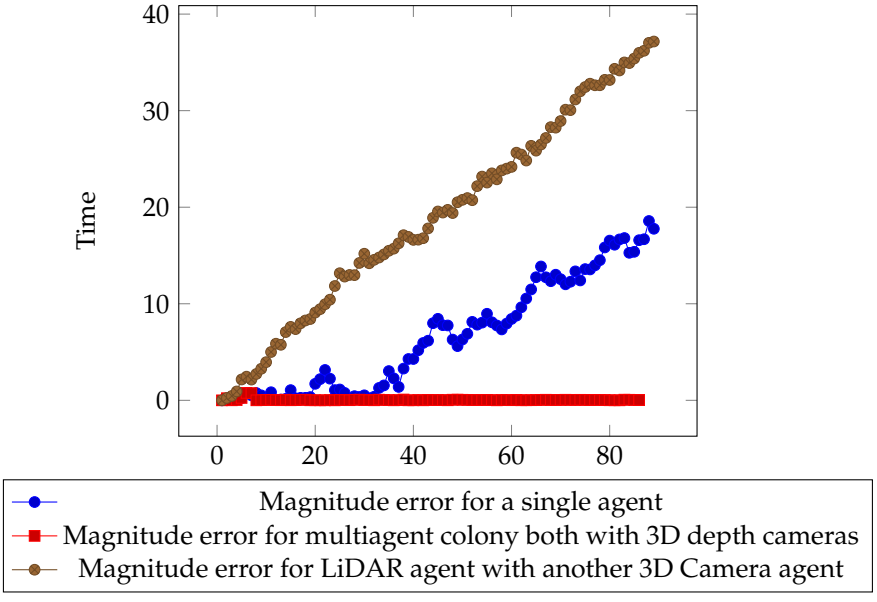

Figure 30. Data showing error over time for all sets of errors.

Then, we examine the error when another agent is sharing environment information. Figure 31 clearly shows the error starting to rise; however, after reaching a quick peak, communication with another local agent is made and the pose error drops significantly. From this point on, the error does not grow beyond a concerning value.

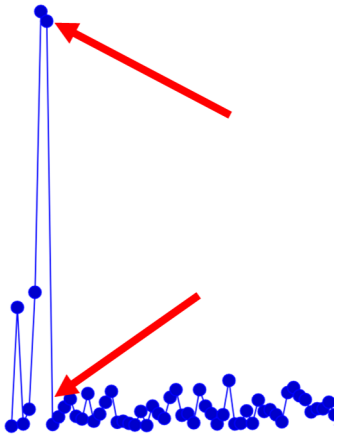

Figure 31. Image indicating the sharp rise and fall in the error with a multiagent configuration.

4. Conclusions

In conclusion, this paper proposes a method for decentralised robotic agents to map their environment using a voxel grid. This approach enables the recording of data over time in a fixed space, improving the certainty of an object's location as the agents explore, without being invalidated by dynamic or moving objects. However, it is important to note that this method does not replace SLAM, as it still relies on landmark discovery and agent pose alignment.

While the benefits of this approach have not been fully evaluated, it provides a solid foundation for future work. For instance, further research can investigate the impact of chaotic landmarks on the performance of the proposed method. Additionally, the data stored in the voxel grid can be leveraged across multiple environments and robotic colonies. Overall, this method shows promising results and offers exciting opportunities for future developments in this area.

Author Contributions: Conceptualization, S.B. and A.G.; methodology, A.G.; software, S.B.; validation, A.G., J.I. and Y.C.; formal analysis, S.B.; investigation, S.B.; resources, S.B.; data curation, S.B. and A.G.; writing—original draft preparation, S.B.; writing—review and editing, A.G., J.I. and Y.C; visualization, S.B.; supervision, A.G. and J.I.; project administration, Y.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Fax, J.; Murray, R. Information Flow and Cooperative Control of Vehicle Formations. *IEEE Trans. Autom. Control* **2004**, *49*, 1465–1476. [CrossRef]
2. Lončar, I.; Babić, A.; Arbanas, B.; Vasiljević, G.; Petrović, T.; Bogdan, S.; Mišković, N. A heterogeneous robotic swarm for long-term monitoring of marine environments. *Appl. Sci.* **2019**, *9*, 1388. [CrossRef]
3. Schmuck, P.; Scherer, S.A.; Zell, A. Hybrid Metric-Topological 3D Occupancy Grid Maps for Large-scale Mapping. *IFAC-PapersOnLine* **2016**, *49*, 230–235. [CrossRef]
4. Schmuck, P.; Chli, M.; Schmuck, P. CCM-SLAM: Robust and efficient centralized collaborative monocular simultaneous localization and mapping for robotic teams CCM-SLAM: Robust and Efficient Centralized Collaborative Monocular SLAM for Robotic Teams. *J. Field Robot.* **2019**, *36*, 763–781. [CrossRef]

5. Jadbabaie, A.; Jie, L.; Morse, A. Coordination of groups of mobile autonomous agents using nearest neighbor rules. *IEEE Trans. Autom. Control* **2003**, *48*, 988–1001. [CrossRef]
6. Lin, Z.; Broucke, M.; Francis, B. Local Control Strategies for Groups of Mobile Autonomous Agents. *IEEE Trans. Autom. Control* **2004**, *49*, 622–629. [CrossRef]
7. Andrade-Cetto, J.; Sanfeliu, A. The Effects of Partial Observability in SLAM. In Proceedings of the IEEE International Conference on Robotics and Automation, New Orleans, LA, USA, 26 April–1 May 2004.
8. Mur-Artal, R.; Montiel, J.M.; Tardos, J.D. ORB-SLAM: A Versatile and Accurate Monocular SLAM System. *IEEE Trans. Robot.* **2015**, *31*, 1147–1163. [CrossRef]
9. Campos, C.; Elvira, R.; Rodriguez, J.J.; Montiel, J.M.; Tardos, J.D. ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual-Inertial, and Multimap SLAM. *IEEE Trans. Robot.* **2021**, *37*, 1874–1890. [CrossRef]
10. Wang, G.; Gao, X.; Zhang, T.; Xu, Q.; Zhou, W. LiDAR Odometry and Mapping Based on Neighborhood Information Constraints for Rugged Terrain. *Remote Sens.* **2022**, *14*, 5229. [CrossRef]
11. Yuan, M.; Li, X.; Cheng, L.; Li, X.; Tan, H. A Coarse-to-Fine Registration Approach for Point Cloud Data with Bipartite Graph Structure. *Electronics* **2022**, *11*, 263. [CrossRef]
12. Muglikar, M.; Zhang, Z.; Scaramuzza, D. Voxel Map for Visual SLAM. In Proceedings of the 2020 IEEE International Conference on Robotics and Automation (ICRA), Paris, France, 31 May 2020–31 August 2020.
13. Siva, S.; Nahman, Z.; Zhang, H. Voxel-based representation learning for place recognition based on 3D point clouds. In Proceedings of the IEEE International Conference on Intelligent Robots and Systems, Las Vegas, NV, USA, 24 October 2020–24 January 2021; pp. 8351–8357. [CrossRef]
14. Bailey, B.Y.T.I.M.; Durrant-whyte, H. Simultaneous Localization and Mapping (SLAM): PART II. *Update* **2006**, *13*, 108–117. [CrossRef]
15. Smith, R.; Self, M.; Cheeseman, P. Estimating Uncertain Spatial Relationships in Robotics. *Mach. Intell. Pattern Recognit.* **1988**, *5*, 435–461. [CrossRef]
16. Leonard, J.; Durrant-Whyte, H. Mobile robot localization by tracking geometric beacons. *IEEE Trans. Robot. Autom.* **1991**, *7*, 376–382. [CrossRef]
17. Saeedi, S.; Trentini, M.; Seto, M.; Li, H. Multiple-Robot Simultaneous Localization and Mapping: A Review. *J. Field Robot.* **2016**, *33*, 3–46. [CrossRef]
18. Saha, A.; Dhara, B.C.; Umer, S.; AlZubi, A.A.; Alanazi, J.M.; Yurii, K. CORB2I-SLAM: An Adaptive Collaborative Visual-Inertial SLAM for Multiple Robots. *Electronics* **2022**, *11*, 2814. [CrossRef]
19. Dube, R.; Gawel, A.; Sommer, H.; Nieto, J.; Siegwart, R.; Cadena, C. An online multi-robot SLAM system for 3D LiDARs. In Proceedings of the IEEE International Conference on Intelligent Robots and Systems, Vancouver, BC, Canada, 24–28 September 2017; pp. 1004–1011. [CrossRef]
20. Zhang, S.; Zheng, L.; Tao, W. Survey and Evaluation of RGB-D SLAM. *IEEE Access* **2021**, *9*, 21367–21387. [CrossRef]
21. Deng, W.; Huang, K.; Chen, X.; Zhou, Z.; Shi, C.; Guo, R.; Zhang, H. Semantic RGB-D SLAM for Rescue Robot Navigation. *IEEE Access* **2020**, *8*, 221320–221329. [CrossRef]
22. Runz, M.; Buffier, M.; Agapito, L. MaskFusion: Real-Time Recognition, Tracking and Reconstruction of Multiple Moving Objects. In Proceedings of the 2018 IEEE International Symposium on Mixed and Augmented Reality (ISMAR), Munich, Germany, 16–20 October 2019; pp. 10–20. [CrossRef]
23. Jiang, J.; Wang, J.; Wang, P.; Chen, Z. POU-SLAM: Scan-to-model matching based on 3D voxels. *Appl. Sci.* **2019**, *9*, 4147. [CrossRef]
24. Cattaneo, D.; Vaghi, M.; Valada, A. LCDNet: Deep Loop Closure Detection and Point Cloud Registration for LiDAR SLAM. *IEEE Trans. Robot.* **2021**, *38*, 2074–2093. [CrossRef]
25. Xiang, H.; Shi, W.; Fan, W.; Chen, P.; Bao, S.; Nie, M. FastLCD: A fast and compact loop closure detection approach using 3D point cloud for indoor mobile mapping. *Int. J. Appl. Earth Obs. Geoinf.* **2021**, *102*, 102430. [CrossRef]
26. Liu, K.; Ou, H. A Light-Weight LiDAR-Inertial SLAM System with Loop Closing. *arXiv* **2022**, arXiv:2212.05743.
27. Lewis, J.; Lima, P.U.; Basiri, M. Collaborative 3D Scene Reconstruction in Large Outdoor Environments Using a Fleet of Mobile Ground Robots. *Sensors* **2022**, *23*, 375. [CrossRef] [PubMed]
28. Luo, J.; Shu, X.; Zhai, Y.; Fu, X.; Ding, B.; Xu, J. A Fast and Robust Solution for Common Knowledge Formation in Decentralized Swarm Robots. *J. Intell. Robot. Syst.* **2022**, *106*, 68. [CrossRef]