IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 2 January 2023, accepted 21 January 2023, date of publication 25 January 2023, date of current version 31 January 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3239657

RESEARCH ARTICLE

Camera Pose Optimization for 3D Mapping

IKER LLUVIA[1], ELENA LAZKANO[2], AND ANDER ANSUATEGI[1]
[1]Autonomous and Intelligent Systems Unit, Tekniker, Basque Research and Technology Alliance (BRTA), Eibar, 20600 Gipuzkoa, Spain
[2]Department of Computer Science and Artificial Intelligence, University of the Basque Country (UPV/EHU), Donostia-San Sebastian, 20018 Gipuzkoa, Spain

Corresponding author: Iker Lluvia (iker.lluvia@tekniker.es)

This work was supported in part by the Project "5R-Red Cervera de Tecnologías Robóticas en Fabricación Inteligente," through the "Centros Tecnológicos de Excelencia Cervera" Program funded by the "Centre for the Development of Industrial Technology (CDTI)," under Contract CER-20211007.

ABSTRACT Digital 3D models of environments are of great value in many applications, but the algorithms that build them autonomously are computationally expensive and require a considerable amount of time to perform this task. In this work, we present an active simultaneous localisation and mapping system that optimises the pose of the sensor for the 3D reconstruction of an environment, while a 2D Rapidly-Exploring Random Tree algorithm controls the motion of the mobile platform for the ground exploration strategy. Our objective is to obtain a 3D map comparable to that obtained using a complete 3D approach in a time interval of the same order of magnitude of a 2D exploration algorithm. The optimisation is performed using a ray-tracing technique from a set of candidate poses based on an uncertainty octree built during exploration, whose values are calculated according to where they have been viewed from. The system is tested in diverse simulated environments and compared with two different exploration methods from the literature, one based on 2D and another one that considers the complete 3D space. Experiments show that combining our algorithm with a 2D exploration method, the 3D map obtained is comparable in quality to that obtained with a pure 3D exploration procedure, but demanding less time.

INDEX TERMS 3D mapping, active vision, exploration, mobile robotics, next best view, ray-tracing.

I. INTRODUCTION

Mobile robots are of great value in many areas, such as logistics, inspection and maintenance, or personal assistance. For this reason, their popularity has grown rapidly in the recent years, being demanded for a wide variety of areas and objectives. Nevertheless, there are still many challenges for an effective and safe autonomous operation [1]. Autonomous navigation is still a challenge due to the specific requirements and peculiarities of each scenario to which robotic systems have to be adapted.

Navigation requires planning the trajectory to the destination, and thus the robot needs knowledge of the morphology of the environment usually in the form of a pre-built map. The mapping process is performed by guiding the robot through the whole environment by an operator by means of teleoperation. The generated map is used for localisation purposes too, to estimate the position of the robot in the world, specially in GPS-denied environments.

Simultaneous localisation and mapping (SLAM) is the problem of building a map with uncertain localisation. Multiple algorithms and techniques have emerged in the last two decades, but most of them share the guidance of the robot by a human during the mapping process to ensure the full coverage of the environment and to ease the detection of loop closures as well. In fact, loop closure detection is the process of recognising an already mapped area, and it is still one of the biggest problems in SLAM [2]. The seed of loop closure is to minimise the accumulated error [3], [4] during mapping and correct the map being built, increasing the coherence between the digital representation and the real scenario. Despite this, the resulting model usually needs a post-processing refining process to correct any erroneously added element. Altogether, it is not a straightforward process to perform all these actions automatically without no human intervention.

The process of teleoperating a robot to map an environment is usually a highly time consuming task, especially in

The associate editor coordinating the review of this manuscript and approving it for publication was Muhammad Sharif.

large areas or when the movement of the robot is limited. In other cases, it is difficult or even impossible for the robot to be guided due to insufficient connectivity or dangerous conditions, such as in rescue operations of natural disasters. Also, a technician should be available and present so the robot completes the mapping process efficiently, which is not always possible in real scenarios. Besides, environments where mobile robots end up operating in are often unstructured and changing, and it leads to a fast decay of the performance over time. This issue makes it necessary to rebuild or edit the map regularly.

Actively calculating robot paths while building the map of an environment without prior information is called Active SLAM (ASLAM) [5]. The *active* term refers to an online search for exploration destinations for mapping, instead of the traditional ''passive'' and teleoperated procedure. ASLAM simplifies the setting up of a navigation system in many applications, as the robot is capable of building the map by itself with no human interaction.

Hitherto, the concept of ASLAM has been introduced from the field of mapping, but a similar problem has been addressed in computer vision. Indeed, the problem was first discussed about 20 years ago by computer vision researchers Bajcsy, Cowan, Kovesi and others, and it was referred as *active sensing*, *active vision* or *active perception* [6], [7]. Active vision is defined as a reactive action of the sensor to obtain information from the environment for a particular purpose. Applied to mobile robotics, active perception refers to the capability of the robot to actively modify the sensor state to get a better understanding of the surroundings and reduce the localisation uncertainty. More specifically, this variant is known as *active localisation* [8].

In general, an ASLAM algorithm consists of three iterative phases [9]:

- Pose identification. Given the partial map of the environment, the robot identifies a set of destinations that will potentially increase the mapped area of the environment. They can also be destinations to reduce both the uncertainty of the localisation and the uncertainty of some already seen elements. The computational complexity of the evaluation grows exponentially with the search space [10]. A concept that is widely used in the context of exploration is that of a *frontier* [11]. Frontiers are free known points next in the boundary of unexplored regions. In principle, they are optimal destinations to which the robot should navigate to expand the explored area.
- Goal selection. Potential destinations are evaluated and the best one selected, estimating the *cost* and *gain* of each of them. Each approach may use a different method, but most of them consider variables such as the distance between the actual position and the target pose or the size of the frontier. Generally, cost and gain are inputs for a final value called *utility* [12]. Utility serves as a metric to compare exploration trajectories [13], [14]. Ideally, to compute the real utility of a given action, the robot should reason about the evolution of the posterior over the robot pose and the map, taking into account future (controllable) actions and future (unknown) measurements. However, computing this joint probability analytically is, in general, computationally intractable, and thus, it is approximated [15].
- Navigation and checking. The robot moves towards the goal selected in the previous step, updating the map in the process, i.e., adding new elements or modifying the already added ones. The robot may navigate using a classic algorithm [16] or an exploration oriented technique [17]. The navigation finishes when (a) the robot reaches the goal, (b) another optimal destination is detected and sent to the navigation module, or (c) a certain termination criteria is satisfied. Until the last case occurs, the system keeps identifying poses and navigating to the best ones iteratively. A termination criteria is required in order the robot not to loop indefinitely [18], [19].

ASLAM methods for the generation of 3D maps have two main drawbacks for their implementation. On the one hand, its computational cost is very high, due to the large number of elements that an unknown environment can have and the many different possibilities of exploring it. On the other hand, depending on the characteristics of the sensor and the structure of the environment, the robot may have to navigate to more destinations than one would expect in order to map the entire space. These two reasons make ASLAM algorithms require a large amount of time to complete the scan. Our objective is to test whether, using the same time as a 2D exploration algorithm, we can obtain a 3D map similar to that obtained by a complete 3D approach. To this end, we propose a method that optimises the pose of a sensor so that it always faces the area of the environment about which it has the least information.

This paper is organised as follows. Related work in the ASLAM literature is reviewed in Section II. The proposed approach is described in Section III. How the presented work is implemented is described in Section IV, explaining the evaluation performed in simulation in Section V. Section VI summarises the paper and provides future work alternatives.

II. RELATED WORK

ASLAM approaches may focus on improving any of the phases described in Section I (*pose identification*, *goal selection* and *navigation and checking*). One relevant aspect that differentiates one algorithm from another is whether it optimises the complete trajectory of the robot during exploration or it only estimates an optimal viewpoint [17], [20]. In the latter case, only the navigation destination is calculated, and a generic algorithm that does not address exploration is in charge of path planning. Another common criterion to group these methods is the localisation uncertainty. Exploration can assume perfect positioning [21] or, on the contrary, it can address localisation uncertainty too [22], setting destinations

not only to broaden the knowledge about the environment, but also to improve localisation estimates. Also, the search space for the pose identification step may vary as well. Candidates can be found in the entire map, or they can be limited to a part of it. Additionally, the candidate evaluation can be performed sequentially before and after the navigation step, or it can be performed continuously while the robot moves through the environment. In this case, the complete trajectory can be fulfilled, or, instead, it can be interrupted and a trajectory towards a new destination executed. As we can observe, the problem of ASLAM can be faced with many different perspectives, and the there are many trends in optimisation. In the following paragraphs, a brief review of the different approaches relevant to the current literature is done.

Even though it is one of the first contributions to ASLAM and not a recent work, we consider that the approach of Brian Yamauchi [11] must be underlined, as many methods are still based on the same strategy. He proposes a solution based on frontiers, the key idea behind which is: ''to gain the most new information, move to the boundary between open space and uncharted territory''. To do so, he uses an evidence grid map where each point is free, occupied or unknown. Free points adjacent to unknown points are potential exploration destinations and they are grouped into frontier regions. Then, the robot navigates iteratively to the nearest reachable frontier. From there, the robot is able to get observations of unexplored space and add them to the map, in addition to seeing new potential goals. The path planner uses a depth-first search to calculate the shortest obstacle-free path from the robot's current position to the goal location. Navigating to each vantage point and discarding inaccessible ones, the robot can map every reachable point in the environment.

Senarathne and Wang [23] extend Yamauchi's work to 3D volumes and present a strategy based on the concept of surface frontiers. They detect surfaces in a given 3D occupancy grid map and extract their edges. Then, voxels that do not have their six faces exposed to unmapped space are discarded and the rest are considered as frontier voxels. Finally, surface frontier representatives are generated that indicate the direction in which the surface is projected. Similarly, Dornhege and Kleiner [24] present a frontier-based ASLAM of a 3D environment, but they seek frontiers in the complete volumetric space, and not only using surfaces. Besides, they set a specific number of poses as a termination criteria.

Some ASLAM methods propose to optimise the trajectory itself for mapping purposes and not only for the exploration goal of each iteration. In this vein, the use of Rapidly Exploring Random Trees (RRTs) [25] with a Receding Horizon (RH) strategy is a well-known sampling-based technique [26]. Umari and Mukhopadhyay [27] use RRTs to grow towards unknown regions and passively detect frontiers. The tree is not used to define the robot trajectory itself, but to search for frontier points. It runs independently of the robot movement. Likewise, Papachristos et al. [28] present an RH-based ASLAM strategy that considers the uncertainty of the robot localisation as well. In fact, the RRT is only used to find exploration destinations, and it is a second planning layer the one that aims to optimise the probabilistic mapping behaviour of the robot and minimise the root's belief uncertainty. In an attempt to get the benefits of different exploration methods, some approaches propose combining them in a single ASLAM solution [10], [29], [30].

As during the exploration a complete map is not available, some approaches concur that ASLAM strategies should include explicit place revisiting actions to reduce the localisation uncertainty [31]. In that vein, Carlone et al. [17] present a method that evaluates the particle-based SLAM posterior approximation using the Kullback–Leibler divergence to decide between exploration and place revisiting. More recently, Lehner et al. [32] integrate this same concept upon a submap-based 6D ASLAM system. Similarly, Valencia and Andrade-Cetto [33] evaluate the utility of exploration and place revisiting sequences to choose the one that minimises the overall map and path entropy. On the other hand, Sadat et. al. [34] consider feature-richness during path planning to direct the sensor of a drone towards high-density areas and avoid visually-poor sections, as the latter can increase the uncertainty in pose estimation and make SLAM fail. The candidate viewpoints are also ranked based on their surface normals and the viewing distance.

Concerning dimensionality, nowadays most of the mobile robots are designed for 2D navigation, albeit there can be slopes, stairs or elevators that make the robot operate at different heights. In these situations, mobile systems use multi-level maps [35] and alternate among different layers, but the motion is still performed in a plane. Here, robot poses, velocity commands, and trajectories include only x and y positional values and rotations around the z axis (*yaw*). Indeed, 3D motion calculations increase the complexity of the problem considerably, and often it is not necessary. In this line, Micro Aerial Vehicles (MAVs) have been widely used in surveillance, search and rescue, exploration and mapping applications. Their reduced size and high manoeuvrability make them ideal for moving in cluttered environments. The algorithms developed in this area go beyond the limitations of ground robots, but there are several interesting approaches to take into consideration. For example, Kompis et al. [36] present an informed sampling approach that takes advantage of surface frontiers to sample viewpoints only where high information gain is expected. Potential Next-Best-Views (NVBs) are sampled from the MAV's configuration space using surface frontiers, and ranked by their expected information gain. Since the computational power of the MAV is limited, they define a heuristic to decide which proposed viewpoint to evaluate next.

Davison and Murray [37], [38] implement a general system for autonomous localisation using active vision,

where a stereo head is controlled in real time during SLAM to improve localisation accuracy. They head the cameras towards certain areas to ensure that persistent features are rematched, reducing the motion drift. To decide which feature the vision system should target, they calculate uncertainties for all visible features, and choose the one with the largest uncertainty. Marchand and Chaumette [39] do consider unknown space in their work, as they propose a 3D scene reconstruction system based on an information gain function that represents either the observation of a new object or the certainty that a given region is object-free. Since the reconstruction is performed via a camera mounted on a robotic arm and it is limited to geometric primitives, they avoid unreachable viewpoints and positions in the vicinity of the robot joint limits. Isler et al. [40] go a step further, and propose mounting the arm on a mobile platform to achieve the complete 3D reconstruction of more complex objects. They employ an information gain function that considers both unknown voxels and their entropies, being these last ones derived from the occupancy probability. Delmerico et al. [41] also propose a set of volumetric information formulations and evaluate them together with recent formulations in the literature [42], [43].

Even though many more different methods have been proposed for active sensor control [44], [45], [46], most of them focus on the optimisation of the handled task once the scans have been added to the map, without intervening in the map building process itself. Instead, these approaches make use of common occupancy grid maps, which may include colour data or not, but they are not optimised for the posterior information estimation of already seen areas. We believe that the performance of 3D reconstruction algorithms is dependant upon the data stored in the map and, for this reason, our proposal includes additional information for uncertainty calculations in the map creation step.

In this work, we present an ASLAM system that optimises the pose of the sensor for the 3D reconstruction of an environment, while a 2D algorithm controls the motion of a mobile platform. A Rapidly-Exploring Random Tree (RRT) [27] algorithm is used for the ground exploration strategy, which calculates the optimal exploration destination and sends it to the navigation unit. Then, the navigation system calculates an obstacle-free trajectory and the mobile platform executes it. Simultaneously, the pose of the camera is optimised based on the state of the 3D map and the position of the platform, in order to capture the most relevant information possible in each iteration. The information gathered from the camera is fed back into the 3D map. The main contributions of this work are as follows:

1) A system to create a 3D model of the environment that provides information about the quality of each mapped area. This model serves as a metric to compare different reconstruction approaches.
2) An active vision method that optimises the pose of a camera to explore the environment and increase the quality of the resulting map.

III. PROPOSED APPROACH

Our approach makes use of active perception to search for viewpoints of the sensor that reveal areas of the environment that increase coverage together with the quality of the map being built. Our method differs from the ones mentioned above in the fact that it relies on the characteristics of the acquisition sensor itself and their effect on the data obtained to calculate the information gain and the NBV, instead of considering the size of the area to be mapped or the localisation uncertainty. In addition, it is proposed as a complementary method to an exploration process, optimising the perception capabilities of the robot used, and not a complete ASLAM method. The proposed approach is based on its own octree-like structure, which makes it independent and compatible with other mapping and navigation methods. Broadly, the system is an iterative algorithm that: captures a frame from the sensor; updates the 3D model according to it; calculates the next best sensor pose; moves the sensor; and repeats the process again capturing a frame from the new pose. In the generated 3D model, an uncertainty value together with spatial and occupancy information are stored for each point. In the next two sections, the uncertainty estimation and the algorithm itself are explained in detail.

A. UNCERTAINTY ESTIMATION

As we want the system to consider not only the completeness or coverage of the map but also the quality, we need a value that represents this aspect. It is hard to define or measure the quality of a map when there is no ground truth to compare it with, that is what happens most of the times in robotic mapping scenarios. In the absence of this option, the quality of a map can be seen as the reliability of its data. As uncertainty is a common term in mobile robotics and it is directly related to the reliability, it has been adopted for the representation built in our method.

The *uncertainty map* presented here is an *octree*-based [47] representation. Octrees are a tree data structures where each branch node has eight children, as it represents an octant. Tree nodes store information regarding the space they represent. In our proposal, leaf nodes are voxels of the size of the resolution of the map, and they include uncertainty values u which are calculated as a function of the pose they have been seen from. Each time a cell is seen, the new value and the stored are combined to update the uncertainty value accordingly. The u value of non-leaf nodes is thus inferred from leaf nodes. Every 3D point observed can be tracked, whether it represents free or occupied space.

We consider that points in the scene at a certain distance are more likely to be correctly captured by the sensor than elements that are closer or farther from it. Cameras are configured to see optimally a specific plane in the scene, called the focal plane. The closer an object is to the focal plane, the sharper it is seen. This attribute is degraded gradually until an element is too far from this location and it is completely blurred. The depth of field is then defined as the distance between the closest and farthest objects in a

photograph that appear to be acceptably sharp [48]. This fact is modelled on our uncertainty map in two ways. On the one hand, from the set of points P captured by the sensor, only the points p whose Euclidean distance to the lens $d(p)$ falls in the range $[D_{min}, \ \ldots \ , D_{max}]$ are added to the map M. Hence, elements that lay outside the depth of field are not taken into account. Therefore, being $p(x, y, z)$ a point whose spacial coordinates in the Euclidean space are x, y and z:

$$p(x, y, z) \in M \iff p(x, y, z) \in P : \{D_{min} < d(p) < D_{max} \mid d(p) = ||\overrightarrow{(x, y, z)}||_2\}$$

On the other hand, the distance $d(p, F)$ from each gathered point $p(x, y, z)$ to the focal plane F is measured as the difference between the distance from the lens to the point $d(p)$ and the distance from the lens to the focal plane D_F:

$$d(p, F) = |d(p) - D_F|, \tag{1}$$

Finally, the distance-based uncertainty of each point p in M is estimated according to $d(p, F)$, as shown in Equation 2.

$$u_d(p, F) = 1 - \frac{2}{1 + e^{d(p,F)}} \tag{2}$$

Equation 2 is a sigmoid function that represents the gradual degradation of the sharpness of the objects as explained above. As it only receives positives values, it does not have its characteristic "S"-shaped curve, as shown in Figure 1.

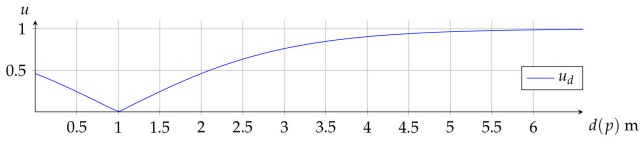


FIGURE 1. Plot of the function used to calculate the uncertainty of a point based on the distance to the lens in metres, being the distance to the focal plane $D_F = 1$.

Similarly, we assume that, given the FOV of a lens, points that lay on the boundary of the frame have a higher uncertainty due to the inherent distortion of the lens [49]. Although there are methods such as camera calibration to correct this deviation, they never reduce the error to zero. To represent this phenomenon, we define a function that calculates an uncertainty value for each point $p(x, y, z) \in P$, with respect to the sensor origin. The angles the point $p(x, y, z)$ forms with the horizontal plane XZ and vertical plane YZ, α_h and α_v respectively, are calculated as follows:

$$\alpha_h = \arctan(\frac{x}{z}) \tag{3}$$

$$\alpha_v = \arctan(\frac{y}{z}) \tag{4}$$

Besides, the horizontal FOV_h and vertical FOV_v angles determined by the sensor itself are required. They are used to perform the unity-based normalisation and bring the angles from Equations 3 and 4 into the range [0, 1]:

$$\alpha'_h = |\frac{\alpha_h}{FOV_h}| \tag{5}$$

$$\alpha'_v = |\frac{\alpha_v}{FOV_v}| \tag{6}$$

Then, the angle-based uncertainty u_α value of a point is the average between α'_h and α'_v:

$$u_\alpha(p) = \frac{\alpha'_h + \alpha'_v}{2} \tag{7}$$

Lastly, distance-based uncertainty u_d obtained from Equation 2 and angle-based uncertainty u_α from Equation 7 are combined into a unique u value, as shown in Equation 8.

$$u(p) = \frac{u_d + u_\alpha}{2} \tag{8}$$

u is the final uncertainty estimation calculated for each point of a scan. Figure 2 shows the impact one function or another has in the uncertainty map built.

Nevertheless, once this value is calculated for an input point, it must be checked whether it is already mapped or not. If not, it is considered as a new unmapped point and it is added to the octree directly with its u value. This action increases the mapped area, i.e. the coverage. On the contrary, if the input point is already known and it has an uncertainty value from a previous iteration, it must be updated instead, enhancing the map quality. Here, we propose to store the uncertainty value that corresponds to the best viewpoint from which a certain point has been captured. That is, if a voxel v is captured for the t-th time, its new uncertainty $u_t(v)$ value is the minimum between the value $u(v_t)$ calculated with Equation 8 and the uncertainty value stored $u(v_{t-1})$ (resultant from the previous iteration):

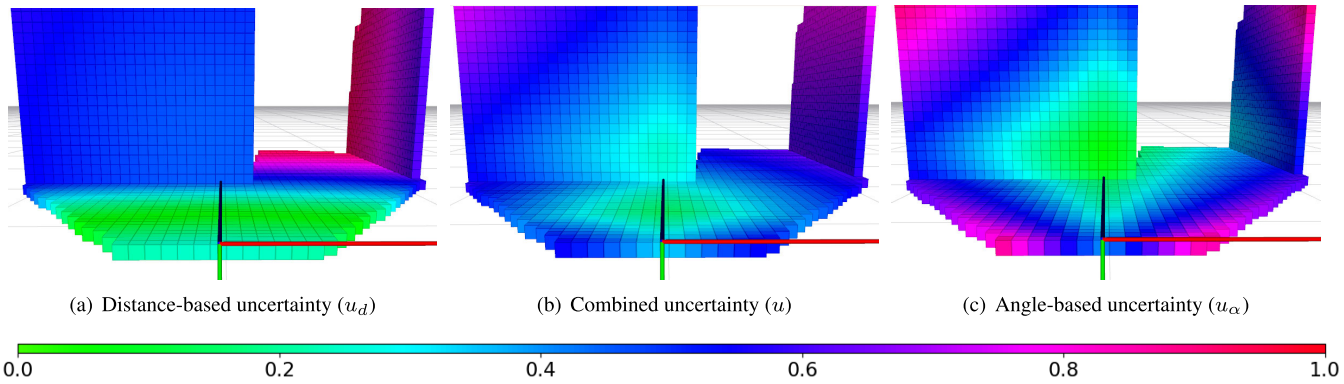
$$u_t(v) = min(u(v_t), u(v_{t-1}) \tag{9}$$

B. CAMERA POSE OPTIMISATION

In this section, the camera pose optimisation algorithm is described, which is in charge of estimating the optimal viewpoint of the camera for a certain state of the uncertainty map. Given a partially completed map and a position on the mobile platform, the optimal pose of the sensor is calculated. The final objective is to maximise the knowledge about the environment. Mapping new areas and improving the reliance of already acquired data can both be considered as increasing the knowledge about the environment, and the uncertainty do represent both aspects. Thus, we can assume that reducing the uncertainty of the map implies progressing towards the final objective.

As the system has no prior knowledge about the environment being mapped, it only stores information about the occupancy value of visited points (nodes), but not about the quality of the exploration itself. So, a new measure is needed to assess if a map is better than another: information.

Information attempts to quantify the amount of knowledge we have about an environment, and it is calculated with the uncertainty value we have already estimated. As Equation 8 maps the inputs into the range (0, 1), it is consistent to say that the uncertainty value that would correspond to unknown cells is 1. Accordingly, their information value can be quantified as 0. Since the information value cannot be



(a) Distance-based uncertainty (u_d) (b) Combined uncertainty (u) (c) Angle-based uncertainty (u_α)

FIGURE 2. Initialisation of the uncertainty map in the same environment for each of the three uncertainty estimation Functions 2, 8 and 7. A reduced HSV colour gradient is used to represent uncertainty values, where the most uncertain voxels are in red and the most reliable ones in green.

negative, the result from the information function i must be inversely proportional to the result of the uncertainty function u. Besides, a cell with uncertainty 0 corresponds to the maximum information. Thus, the information function i must satisfy the following statements:

$$\begin{aligned} &\forall v \in M : i(v) \geq 0 \\ &\forall v, w \in M : \{u(v) \geq u(w) \implies i(v) \leq i(w)\} \\ &\forall v, w \in M : \{u(v) = 0 \implies \nexists w \mid i(w) > i(v)\} \end{aligned}$$

That being said, the result of the information function i is a growing function from 0 to a maximum, as the uncertainty for the same element decreases from 1 to 0. Taking this into account, the proposed information value i for a single cell v is:

$$i(v) = k - u(v) \mid k > 1 \tag{10}$$

k must be greater than 1 to avoid having a negative information value. Considering what information represents, we propose calculating the information of a set of cells or map simply adding the information value of each of the cells inside it. Besides, we present a function whose maximum is the least possible maximum, because greater values would favour the amount of cells in the map. We consider that, at this point, the uncertainty of each cell itself should receive more attention, and that the system can be biased towards exploration in the next-best view selection process. Hence, the information value of a set A is calculated according to Equation 11.

$$I(A) = \sum_{v \in A} (1 - u(v)) \tag{11}$$

The most relevant aspect about the information value of a map is that it allows the comparison between two maps of the same environment in different phases. At any moment t, the system has a particular map M_t built and the camera has a specific pose q_t. The objective is to move the sensor to a pose q_{t+1} such that the map M_{t+1} provides the maximum information possible based on the state of the system in the previous step t. In other words, the pose optimisation algorithm has to maximise the information gain from one iteration to another, moving the sensor accordingly. In addition, the information value I obtained from Equation 11 can also be used to compare maps that, for example, have been built with different approaches.

The challenge here is how to predict the information gain of an unknown point. If a point is already mapped and, thus, its x, y, z coordinates are known, if that point would fall in the sensor's FOV for any of its possible poses can be calculated, together with the exact position at which that point would be seen. Besides, in most of the environments, it is done with a very high accuracy. Using these inputs in the functions presented above, the system can predict how the uncertainty value stored in each cell would be updated and the information gain it would provide, if any. In this case, the information gain of that cell $g(v)$ would be the difference between its actual information value $i(v_t)$ and the hypothetical future value $i(v_{t+1})$. However, when the camera's FOV includes unmapped points, the future estimated information value cannot be calculated mathematically. It is impossible to ensure where the ray would find an object, adding the corresponding cell, and setting it as occupied. To tackle this problem, we propose a "positive" approach, as we imagine that every unknown point will be occupied. The information provided by an unknown point is quantified as 0, i.e., $i(v_t) = 0$, and, with the previous assumption, its information in the potential next iteration $i(v_{t+1})$ is calculated as for any already mapped cell. In summary, the information gain of an unknown point is equal to the information value it has when it is occupied. Mathematically,

$$g(v) = \begin{cases} i(v_{t+1}) - i(v_t) & \text{if } i(v_{t+1}) > i(v_t) \\ 0 & \text{otherwise} \end{cases} \tag{12}$$

Again, being coherent with Equation 11, the information gain of a set of cells A is the sum of the gain of every element inside it:

$$G(A) = \sum_{v \in A} g(v) \tag{13}$$

Once the information gain provided by any future camera pose can be estimated, it is possible to optimise it. The optimisation algorithm simulates the FOV of each pose q

of all the viable poses Q of the sensor, calculating the information gain for each of them. Then, the best one is selected and the sensor is moved accordingly. Taking all calculations into account, the pseudocode of the whole sensor pose optimisation process is shown in Algorithm 1.

Algorithm 1 Camera Pose Optimisation

Require: M : 3D uncertainty map
Require: $SensorFOV$: Sensor's FOV specification
Require: $SensorDOF$: Sensor's movement's degrees of freedom (DOF), ranges and velocities

```
1:  while True do
2:      q_current = GetCurrentPose()
3:      Q = GetPossiblePoses(q_current, SensorDOF, t)
4:      g_max = 0
5:      q_opt = q_current
6:      for each q ∈ Q do
7:          g_q = EstimateGain(q, SensorFOV, M)
8:          if g_q > g_max then
9:              g_max = g_q
10:             q_opt = q
11:         end if
12:     end for
13:     if q_opt ≠ q_current then
14:         MoveSensor(q_opt)
15:     end if
16: end while
```

The proposed camera optimisation process has two key functions that form the core of the algorithm, as they determine the poses to be considered in each iteration and the value - or ranking - assigned to each of them, in order to finally choose the one to be adopted by the sensor. These functions are *GetPossiblePoses* and *EstimateGain* (Line 3 and 7 of Algorithm 1, respectively).

1) GetPossiblePoses

The candidate optimisation poses in each iteration are calculated based on: the sensor's current position, $q_{current}$; its degrees of freedom (DOF), ranges and velocity limits, $SensorDOF$; and a t value, which determines the maximum time the sensor can be in motion between one pose and another, i.e., the maximum time it can take to reach the estimated optimal pose. While the first two parameters are strictly necessary to estimate new poses, the t parameter is added for two main reasons. First, since the platform continues navigating while the sensor motion calculations are performed, the sensor pose selected with respect to the world and the one reached may differ. This difference will be larger the longer the time between sensor poses and the higher the platform speed. Thus, a balance between these two values needs to be maintained. Second, larger values of t imply more possible poses to consider, leading to increased computation time. Note that the elapsed time between consecutive sensor poses is the sum of the computation time plus the sensor movement time, and the t value affects both, contributing to reduce the difference between the initial state of the environment and the final state of each iteration. In this line, there is also a configuration parameter q_{step} that determines the step between poses, which serves not only to discretise the sensor positions, but also to modify the number of them and, consequently, the computational cost of the optimisation. The pseudocode of the complete procedure is shown in Algorithm 2.

Algorithm 2 GetPossiblePoses

Require: q : Relative pose of the sensor
Require: $SensorDOF$: Sensor's movement's degrees of freedom (DOF), ranges and velocities
Require: t : Maximum movement time (0 = no limit)

```
1:  Q = {q}
2:  if t = 0 then
3:      q_ranges = SensorDOF_ranges
4:  else
5:      q_ranges = GetRanges(q, SensorDOF, t)
6:  end if
7:  n = GetNumberOfPoses(q_ranges, q_step)
8:  for i in [1..n] do
9:      q_candidate = GetPose(q_current, q_step, i)
10:     add q_candidate to Q
11: end for
12: return Q
```

2) EstimateGain

Each of the candidate sensor poses Q must be evaluated in order to estimate the optimal one. To this end, a ray-tracing technique is used to predict what the camera would capture in each of the poses. Based on the FOV, range and resolution of the sensor, a virtual ray is cast from the pose of the sensor through each pixel to determine what is visible along the ray in the 3D scene. If the ray reaches the maximum range without hitting any non-free voxel, a gain value of 0 is assigned to it. On the contrary, if it hits a non-free voxel, either occupied (mapped) or unknown (unmapped), the corresponding gain is calculated according to Equation 12. Then, the values of the rays with the same FOV are summed up to get the information gain of the corresponding sensor pose. Repeating this procedure for all the candidate poses, as in Algorithm 1, the optimal pose is obtained, which is sent to the sensor motion module. The optimisation algorithm has no termination criteria, as it is intended to run in parallel and independently of the rest of the mobile robot system.

IV. DESCRIPTION OF THE IMPLEMENTED SYSTEM

Our algorithm is implemented in the well-known Robot Operating System (ROS) framework [50], which is accessible to many other developers and users. Besides, it provides the tools to integrate our functionalities into a robotic platform.

For the experiments presented in this paper, we have integrated our algorithm in a mobile robot. More specifically,

we have used a modified Turtlebot3 Waffle platform. It is a small omnidirectional wheeled robot with a fixed RGB-D camera at the front and a lidar in the centre with 360° vision. To test our camera optimisation algorithm correctly, a telescopic arm with a RGB-D camera on top of it has been added to the platform, which makes possible to pan, tilt, and move the sensor along the vertical axis independently, as shown in Figure 3. However, as the objective is to see the viability of the proposed approach, in these first trials, the optimisation has been limited to the pan movement, i.e., one degree of freedom.

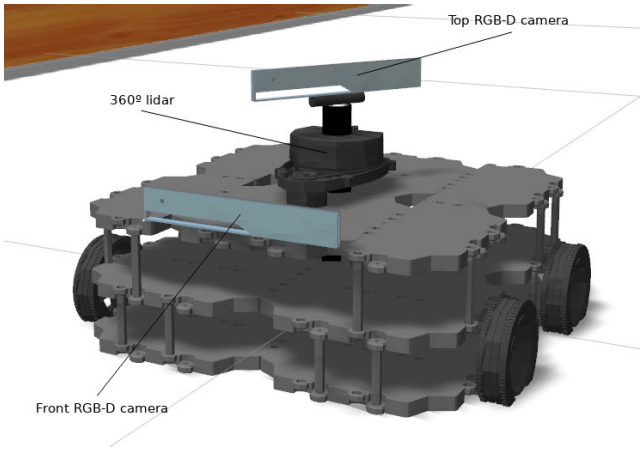


FIGURE 3. Turtlebot3 Waffle platform with a second RGB-D mobile camera added on top of it. In the image, it is rotated 60° left, as it has a 360° pan range.

As our sensor pose optimisation algorithm attempts to be modular and generic without being restricted to a certain type of robot, it does not control the motion of the platform, and publicly available ROS packages are used for this purpose. Our 3D exploration is combined with a package that implements an RRT-based 2D exploration algorithm [27], which finds exploration goals in the horizontal plane and sends them to the navigation system. These destinations are managed by the ROS Navigation Stack, configured for obstacle-free trajectory planning and execution. Thus, while a mobile robot exploration system controls the motion of the platform and builds a 2D map, our system optimises the pose of the camera on the fly to build a dense 3D map. Although both systems run in parallel, they are independent and there is no communication between them.

With this configuration, where the camera pose optimisation algorithm does not take into account the motion of the robot for the camera poses, an adequate t value must be set as it has a significant impact. Essentially, it reduces the main issues of the lack of coordination between platform and camera motion. As the sensor is attached to the platform, when the latter moves, the former moves implicitly in the same direction. As a consequence, the sensor is not in the global pose the optimisation algorithm decided to move to. This effect may be exacerbated when the platform moves faster or the system needs more time to estimate the optimal camera pose. Setting a low t value makes the transformations between consecutive poses smaller and their overlapping regions larger, which eases the 3D map merging [51]. If the same camera is used for localisation, it also improves position estimation [52].

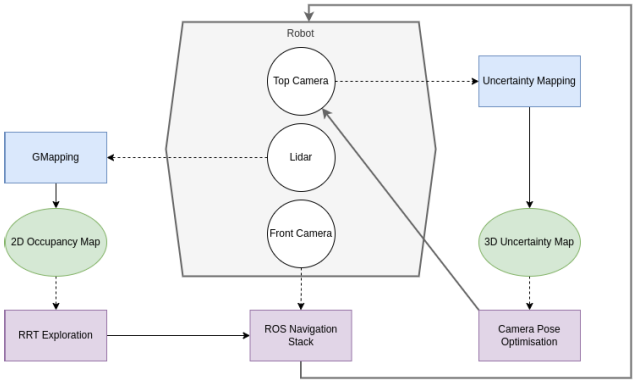


FIGURE 4. Complete ASLAM system's diagram implemented in the mobile robot.

As described before, the uncertainty map simply gets the colour image and the point cloud provided by the top RGB-D camera and creates a 3D octomap accordingly. As mentioned above, the implementation has been done on the standard octomap implementation, and a detailed explanation can be found in the original paper [53]. Apart from occupancy and colour, our version adds the possibility to store uncertainty values with the functions described before. We have also implemented the corresponding Rviz plugins to visualise these values.[1]

In brief, the implementation for testing consists of three independent systems that live together in the same mobile robot: 2D mapping, uncertainty 3D mapping and navigation. They gather information of the environment from different sensors and may communicate with each other using ROS. Besides, they can send commands to the platform and control its motion, which closes the cycle and enables a fully autonomous behaviour. Figure 4 shows a general overview of the logic of the system and its communications.

V. SYSTEM EVALUATION

The system has been tested in diverse simulated environments where the size, the number of elements and the level of detail of these vary. The models used for the experiments are the following:

- ''simple house''. It is provided by the Robotis group in the official manual of the TurtleBot3 platform, in the simulation section (https://emanual.robotis.com/docs/en/platform/turtlebot3/simulation/). It contains six rooms, with a few basic elements, covering a total area of 15 × 10 m. It has been chosen for its simplicity.

[1]The code of the optimisation algorithm and the octomap related functionalities is available online here: https://github.com/fundaciontekniker/aslam-system

- ''apartment''. It is also crated by the AWS Robotics team, and it is a very detailed and realistic representation of an apartment of 19 × 11 m. It is available here: https://github.com/aws-robotics/aws-robomaker-small-house-world. It offers a hint of how the mapping system would perform in a domestic environment.
- ''bookstore''. It is a 3D model of a bookstore developed by the AWS Robotics team. It has many and very detailed elements, such as furniture and books, which makes it a suitable environment to test a 3D mapping system. It can be acquired from: https://github.com/aws-robotics/aws-robomaker-bookstore-world. The size of the bookstore is 15 × 14 m.

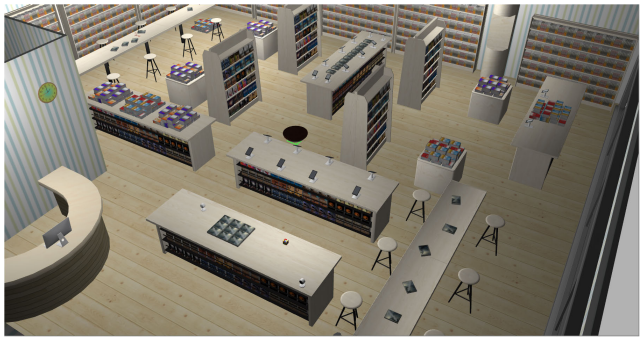


FIGURE 5. ''Bookstore'' test environment.

- ''cafe''. It is a spacious area of 25 × 10 m, with detailed elements along the walls but just a few tables in the middle. This configuration allows the robot to build the 2D map without navigating too much, because the lidar covers nearly all the area from any location. As the FoV of the camera has a reduced scope, the objective of the tests in this environment is to check if our system makes a significant difference in the resulting 3D map in this kind of scenarios. Indeed, the range of lidar for 2D mapping is set to 10 m with a 360° view, while the camera has a 4 m range and a horizontal angle of 59°. Besides, two people are walking through the cafe, adding dynamism to the problem. Here, the behaviour of the system is also evaluated in the presence of dynamic elements. The ''cafe'' model can be obtained from: https://automaticaddison.com/how-to-load-a-world-file-into-gazebo-ros-2/.

FIGURE 6. ''Cafe'' test environment.

- ''warehouse''. It is the usual warehouse of the majority of the factories nowadays. It is another model created by the AWS Robotics team (https://github.com/aws-robotics/aws-robomaker-small-warehouse-world), and it is useful to make an idea of the result the system may obtain in an industrial use case. Its dimensions are 14 × 21 m.
- ''willowgarage''. It is a model available in Gazebo simulator itself by default, already used in many research works. It represents a set of rooms in an office-like area without objects, just walls. It is the largest environment tested, with an area of 65×45 m. As our algorithm solves a local optimisation problem, the size of the environment does not increase the complexity of each iteration. But, it does the size of the 3D uncertainty map being built. It is relevant to test our system in such a large area to see if the results obtained in small environments are extrapolable.

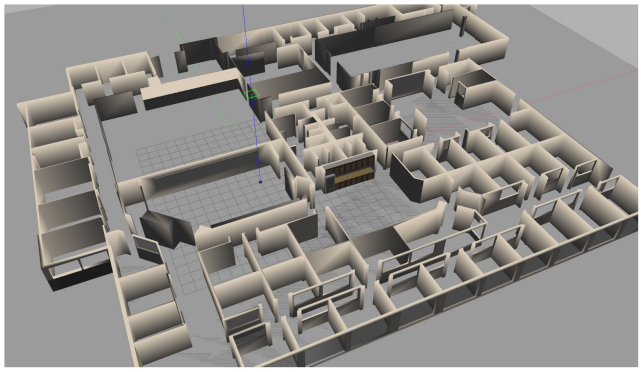


FIGURE 7. ''Willowgarage'' test environment.

Note that all the models used in the tests presented here are publicly available and compatible with the Gazebo simulator. Besides, they are all indoor environments where the mobile platform navigates through a unique flat floor.

In all the environments, the RGB-D sensor simulated is similar to the Inter RealSense R200 camera, with a vertical FOV of 46°, horizontal FOV of 59°, a maximum range of 4 m and a 480 × 360 depth resolution. On the other hand, the 360° lidar is simulated with a range of 100 m and 0.5° of horizontal resolution, which provides 720 points in each scan. Nevertheless, the 2D mapping system is limited to a range of 4 m in all the environments but in ''willowgarage'', where it is limited to 10 m. Larger ranges increase the computational cost of the 2D exploration system, however, 4 m is too low for this environment, because many times, the sensor does not find any obstacles, the localisation has nothing to match, and the robot gets lost. This beam cropping is done for efficiency purposes, as there are no such large free obstacle distances in the selected scenarios.

A. EXPERIMENTAL PROCEDURE

The system has been compared with two different exploration methods from the literature, one based on 2D [27] and another one that considers the complete 3D space [54], [55]. Both

TABLE 1. Comparison of the effectiveness of the obtained 3D maps considering all tests. The metrics shown are: number of voxels, increase in the number of voxels compared to the 2D test in percentage terms, information provided, information gain compared to the 2D test, information gain compared to 2D test in percentage terms.

ENVIRONMENT	TEST TYPE	VOXELS	VOXELS %	INFO	GAIN	GAIN %
simple house (15x10)	2D	63,894		38,898		
	2D+optimisation (t=0.2)	73,291	14.71%	42,705	3,808	9.79%
	2D+optimisation (t=0.5)	78,217	22.42%	46,386	7,489	19.25%
	2D+optimisation (t=1.0)	77,007	20.52%	45,984	7,086	18.22%
	3D	**79,248**	**24.03%**	**46,967**	**8,070**	**20.75%**
apartment (19x11)	2D	86,020		45,664		
	2D+optimisation (t=0.2)	113,218	31.62%	60,465	14,801	32.41%
	2D+optimisation (t=0.5)	118,294	37.52%	63,442	17,778	38.93%
	2D+optimisation (t=1.0)	**131,533**	**52.91%**	**70,747**	**25,083**	**54.93%**
	3D	106,592	23.91%	68,271	22,607	49.51%
bookstore (15x14)	2D	114,651		70,770		
	2D+optimisation (t=0.2)	130,635	13.94%	79,911	9,141	12.92%
	2D+optimisation (t=0.5)	137,217	19.68%	84,014	13,244	18.71%
	2D+optimisation (t=1.0)	139,970	22.08%	85,238	14,467	20.44%
	3D	**153,125**	**33.56%**	**91,666**	**20,896**	**29.53%**
cafe (25x10)	2D	69,566		35,585		
	2D+optimisation (t=0.2)	90,968	30.77%	48,345	12,759	35.86%
	2D+optimisation (t=0.5)	103,998	49.50%	55,636	20,051	56.35%
	2D+optimisation (t=1.0)	102,414	47.22%	56,058	20,473	57.53%
	3D	**136,490**	**96.20%**	**74,196**	**38,611**	**108.50%**
warehouse (14x21)	2D	158,786		85,558		
	2D+optimisation (t=0.2)	177,206	11.60%	95,257	9,700	11.34%
	2D+optimisation (t=0.5)	**193,576**	**21.91%**	104,291	18,734	21.90%
	2D+optimisation (t=1.0)	190,987	20.28%	104,012	18,454	21.57%
	3D	186,610	17.52%	**107,573**	**22,015**	**25.73%**
willowgarage (65x45)	2D	835,622		461,921		
	2D+optimisation (t=0.2)	1,045,753	25.15%	556,748	94,827	20.53%
	2D+optimisation (t=0.5)	1,129,001	35.11%	610,689	148,767	32.21%
	2D+optimisation (t=1.0)	1,144,487	36.96%	617,163	155,242	33.61%
	3D	**1,371,774**	**64.16%**	**762,823**	**300,902**	**65.14%**

algorithms are based on RRT mainly because it is one of the best performing techniques in exploration. We believe that, by using methods with the same basis, our algorithm gets more isolated and its effect is more clearly observable. The tests are as follows:

- **2D**. In each environment, the 2D autonomous exploration is performed first using a strategy based on the use of multiple RRTs, proposed by Umari and Mukhopadhyay.[2] It calculates the trajectories to explore the entire area and builds the corresponding map using a 360° lidar. During this phase, the RGB-D camera builds a 3D uncertainty map, while the robot moves through the environment keeping the camera pose constant. While exploration is performed, the entire trajectory of the robot is stored in order to replicate it in subsequent tests. In this way, the robot can keep the same pose and velocities in the same timestamps among different tests of the same environment. This allows to compare under equal conditions this first 3D map with results where the 3D camera pose optimisation is performed.
- **2D+optimisation**. After a 2D exploration where a 3D map is built keeping the camera static, the same process is repeated with our pose optimisation algorithm running. As the trajectory executed in these tests is the one recorded in the previous case, the time needed to perform the exploration is exactly the same. However, as the pose of the camera with respect to the robot is continuously modified, the 3D map obtained differs. This type of test has been performed with three different configurations, where the t value varies between 0.2 s, 0.5 s and 1 s.
- **3D**. Finally, the algorithm proposed by the Robust Field Autonomy Lab at Stevens Institute of Technology[3] that performs complete 3D exploration of complex environments by means of a ground mobile robot is executed in the same environment. As this system calculates the trajectories of the mobile platform itself to perform the 3D exploration, its positions may be different from those achieved in the rest of the tests. In this last case, as in the first 2D exploration, the camera is static during execution.

In all tests, the 2D map for trajectory planning and navigation is built using OpenSLAM's GMapping algorithm, which is a Rao-Blackwellized particle filer to learn grid maps from laser range data [56]. These five procedures - one 2D exploration, three 2D exploration with our optimisation, one 3D exploration - are repeated three times in all the six environments described above, with three different starting points in each of them, making a total of 90 experiments. At the end of each of them, the time needed, the number of voxels of the 3D map and the total information provided by

[2]https://github.com/hasauino/rrt_exploration

[3]https://github.com/RobustFieldAutonomyLab/turtlebot_exploration_3d

TABLE 2. Efficiency comparison of the obtained 3D maps considering all tests. The metrics shown are: duration of the exploration, increase in the time needed compared to 2D test in percentage terms, average number of mapped voxels per second, average information mapped per second.

ENVIRONMENT	TEST TYPE	DURATION (s)	TIME %	VOXEL / S	INFO / S
simple house (15x10)	2D	431		148	90
	2D+optimisation (t=0.2)	431	0.00%	170	99
	2D+optimisation (t=0.5)	431	0.00%	**181**	**108**
	2D+optimisation (t=1.0)	431	0.00%	179	107
	3D	1,668	287.01%	48	28
apartment (19x11)	2D	378		228	121
	2D+optimisation (t=0.2)	378	0.00%	300	160
	2D+optimisation (t=0.5)	378	0.00%	313	168
	2D+optimisation (t=1.0)	378	0.00%	**348**	**187**
	3D	3,018	698.32%	35	23
bookstore (15x14)	2D	497		231	142
	2D+optimisation (t=0.2)	497	0.00%	263	161
	2D+optimisation (t=0.5)	497	0.00%	276	169
	2D+optimisation (t=1.0)	497	0.00%	**282**	**172**
	3D	2,041	310.66%	75	45
cafe (25x10)	2D	227		306	157
	2D+optimisation (t=0.2)	227	0.00%	400	213
	2D+optimisation (t=0.5)	227	0.00%	**457**	245
	2D+optimisation (t=1.0)	227	0.00%	451	**247**
	3D	2,274	900.44%	60	33
warehouse (14x21)	2D	446		356	192
	2D+optimisation (t=0.2)	446	0.00%	398	214
	2D+optimisation (t=0.5)	446	0.00%	**434**	**234**
	2D+optimisation (t=1.0)	446	0.00%	429	233
	3D	4,477	904.49%	42	24
willowgarage (65x45)	2D	1,615		517	286
	2D+optimisation (t=0.2)	1,615	0.00%	647	345
	2D+optimisation (t=0.5)	1,615	0.00%	699	378
	2D+optimisation (t=1.0)	1,615	0.00%	**709**	**382**
	3D	40,271	2393.05%	34	19

this one are calculated. The numbers shown in this article are average values.

B. RESULTS

Table 1 shows the comparison of 3D mapping results with the three different systems explained before. As can be seen, there is an improvement in the obtained 3D map by performing a 2D exploration with our optimisation algorithm compared to not using it. Our approach has been tested with three different values of t, and all three configurations help to build a denser and richer 3D model of any of the environments. Note that the value t limits the maximum time the sensor has to move from one pose to the next in each iteration. Thus, greater values evaluate more pose candidates, increasing the optimisation possibilities, at the expense of higher computational cost. For this reason, larger values t generally get slightly better results in the tests. Although, based on the experiments, the optimal t value seems to be relative to the environment being explored.

Surprisingly, in some cases, the combination of a 2D exploration with our camera pose optimisation algorithm gets better results than the 3D exploration system. This is the case of the tests carried out in ''warehouse'' and ''apartment'' environments, obtaining a map with more voxels in the former and one with both more voxels and information in the latter (see Figure 9). It is worth mentioning that in some tests, the 3D exploration algorithm obtains worse result than even

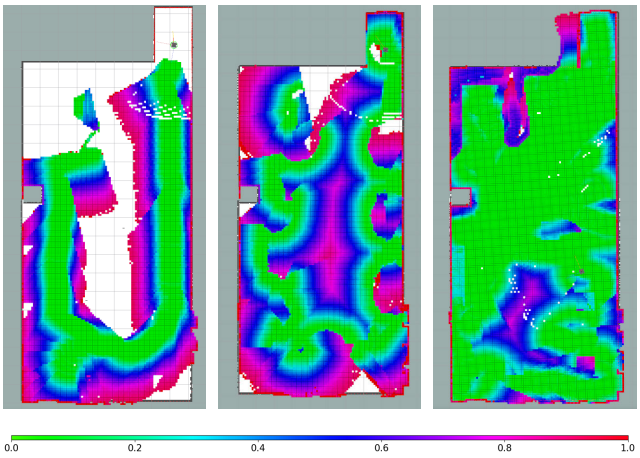


FIGURE 8. ''Cafe'' environment mapping result after 2D basic exploration (left), combination of 2D exploration and camera pose optimisation with $t = 0.5$ (middle) and 3D exploration (right). The 3D uncertainty map is shown with colours, while the 2D map is in greyscale. A reduced HSV colour gradient is used to represent uncertainty values, where the most uncertain voxels are in red and the most reliable ones in green.

the basic 2D exploration because it does not visit some narrow areas where the 2D algorithm does and, indeed, the robot is able to map them. On the contrary, in the ''cafe'' environment, the 2D algorithm is able to map the entire environment with a short trajectory. As a consequence, the camera can only gather information from a small proportion of the environment. As the 3D exploration system calculates its own trajectory

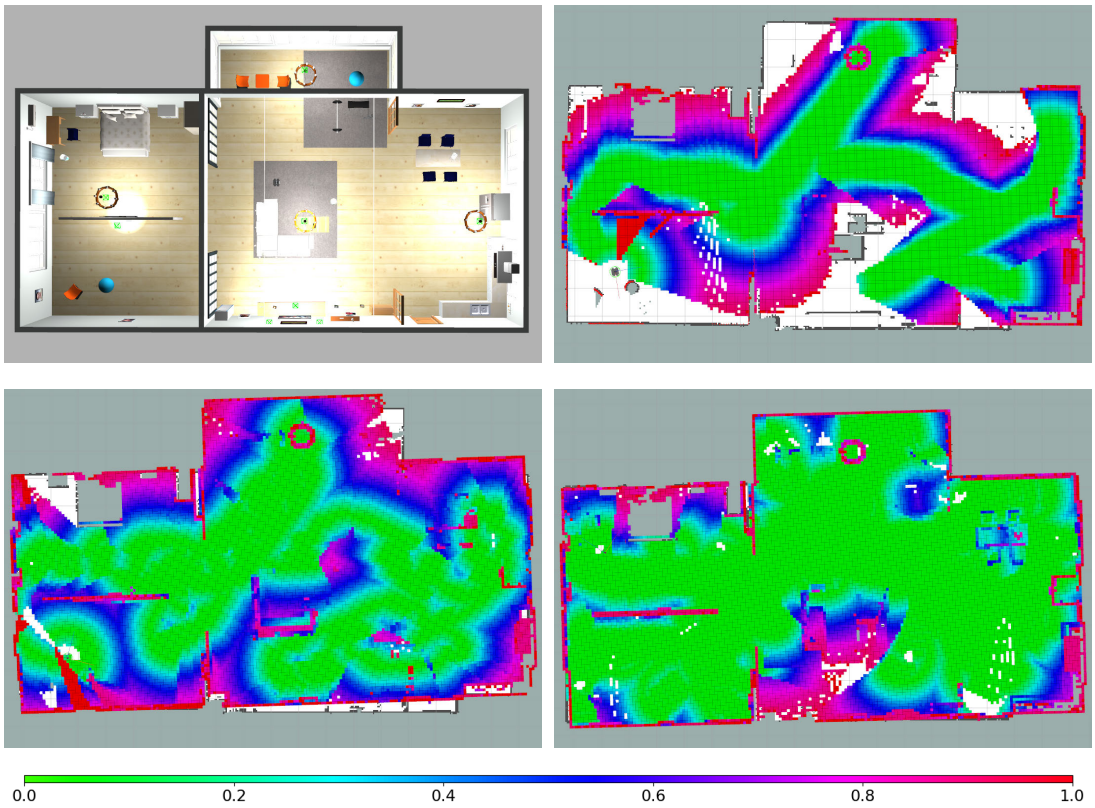


FIGURE 9. "Apartment" environment (top left), mapping result after 2D basic exploration (top right), combination of 2D exploration and camera pose optimisation with $t = 0.5$ (bottom left) and 3D exploration (bottom right). The 3D uncertainty map is shown with colours, while the 2D map is in greyscale. A reduced HSV colour gradient is used to represent uncertainty values, where the most uncertain voxels are in red and the most reliable ones in green.

for this purpose, it maps the whole area and there is a huge improvement in the resulting map. Figure 8 shows the 3D uncertainty maps obtained with the three different systems in the ''cafe'' test environment.

In 5 of the 6 environments, the map with the largest amount of information is achieved with the 3D scanning system, which is quite logical as it is the only procedure of those evaluated here that focuses entirely on that purpose. While the resulting model may differ from one scenario to another, the 3D exploration needs more time to complete the mapping process in all the tests, with an average duration of ten times the duration of the other exploration processes. So, although this system achieves the best result in most cases, the difference is not proportional to the time needed to do so. In this aspect, the combination of 2D exploration with our camera pose optimisation algorithm is the most efficient in all cases, regardless of the value t used. Table 2 shows how this procedure obtains the best ratio between both the number of voxels and the information value of the resulting model and the time needed to build it. More specifically, in the tests carried out, t values of 0.5 or 1.0 are those that achieve the optimal result.

VI. CONCLUSION AND FUTURE WORK

In this work, we present a system that associates uncertainty values with the mapped points based on where they have been captured, and an algorithm that exploits this information to optimise the sensor pose for exploration. We use an octree-based 3D uncertainty map to estimate the camera's NBV online, which is built during motion. Adding this algorithm to a 2D exploration, a denser and richer model is acquired, without requiring additional time for it. Besides, according to the measures used, it has demonstrated to obtain similar of even better results to a complete 3D exploration system, in a much shorter time. In accordance with these preliminary results, the next immediate step is to test the algorithm in a real scenario. The foresee of how the dynamic world and other well known issues such as odometry drift, image frame rate and lighting conditions, to mention some, can affect the performance of the system remain as further work.

Beyond this, the system is highly configurable and adaptable to different sensors. However, this first version of the algorithm has been implemented with some limitations.

On the one hand, we want to add more degrees of freedom to the camera pose optimisation. In these tests, only the pan movement has been considered, but our goal is to also optimise the tilt and height coordinates in every iteration. However, due to the high computational cost of such optimisation, we need to parallelise the code first taking advantage of the increasing computing power of multicore architectures and lowering the computational cost of the optimisation process.

On the other hand, we want to calculate the angle-based uncertainty taking into account not only the direction of the camera, but also the normal of the corresponding point or surface. Besides, we would like to take into account the trajectory that the robot is performing in order to calculate the NBV. Our system runs independent from the robot motion to make it more generic and because the robot used is omnidirectional, it does not need to rotate to reach any objective. Nevertheless, with other motion systems, there can be a big divergence between the expected NBV and the point of view achieved, and taking the trajectory into account and going one step ahead would make a big difference in these situations.

In addition, running 3D exploration after 2D exploration with sensor pose optimisation finishes could be interesting. In principle, the 3D map obtained should be similar to running the 3D scan directly, but the time required for each of them should not. We would like to evaluate whether the combination performs the whole process in less time. Moreover, using other functions to calculate the uncertainty will lead to different results. Seeing how these affect not only to the map, but also to the localisation, can guide us to a better solution.

ACKNOWLEDGMENT

The authors would like to thank the Autonomous Intelligent Systems Laboratory of Prof. Wolfram Burgard with the Department of Computer Science, University of Freiburg, as a part of this research was done during an internship there. They also thank Tim Caselitz and Michael Krawez for sharing their knowledge and expertise with them.

REFERENCES

[1] M. B. Alatise and G. P. Hancke, "A review on challenges of autonomous mobile robot and sensor fusion methods," *IEEE Access*, vol. 8, pp. 39830–39846, 2020.
[2] E. Garciafidalgo and A. Ortiz, "iBoW-LCD: An appearance-based loop-closure detection approach using incremental bags of binary words," in *Proc. Int. Conf. Robot. Autom.*, May 2018, vol. 3, no. 4, pp. 3051–3057.
[3] W. Hess, D. Kohler, H. Rapp, and D. Andor, "Real-time loop closure in 2D LIDAR SLAM," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2016, pp. 1271–1278.
[4] X. Gao, R. Wang, N. Demmel, and D. Cremers, "LDSO: Direct sparse odometry with loop closure," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2018, pp. 2198–2204.
[5] I. Lluvia, E. Lazkano, and A. Ansuategi, "Active mapping and robot exploration: A survey," *Sensors*, vol. 21, no. 7, p. 2445, 2021.
[6] R. Bajcsy, "Active perception," *Proc. IEEE*, vol. 76, no. 8, pp. 966–1005, Aug. 1988.
[7] R. Bajcsy, Y. Aloimonos, and J. Tsotsos, "Revisiting active perception," *Auto. Robots*, vol. 42, pp. 177–196, May 2018.
[8] Z. Zhang, "Active robot vision: From state estimation to motion planning," Ph.D. thesis, Dept. Bus., Econ. Inform., Univ. Zurich, Zürich, Switzerland, Univ. Zurich, 2020.
[9] M. L. R. Arevalo, "On the uncertainty in active SLAM: Representation, propagation and monotonicity," Ph.D. thesis, Dept. Comput. Sci. Syst. Eng., Univ. Zaragoza, Zaragoza, Spain, 2018.
[10] M. Selin, M. Tiger, D. Duberg, F. Heintz, and P. Jensfelt, "Efficient autonomous exploration planning of large-scale 3-D environments," *IEEE Robot. Autom. Lett.*, vol. 4, no. 2, pp. 1699–1706, Apr. 2019.
[11] B. Yamauchi, "A frontier-based approach for autonomous exploration," in *Proc. IEEE Int. Symp. Comput. Intell. Robot. Autom. CIRA. 'Towards New Comput. Princ. Robot. Automation'*, Jul. 1997, pp. 146–151.
[12] D. Joho, C. Stachniss, P. Pfaff, and W. Burgard, "Autonomous exploration for 3D map learning," in *Autonome Mobile Systeme*. Berlin, Germany: Springer, 2007, pp. 22–28.
[13] H. Carrillo, I. Reid, and J. A. Castellanos, "On the comparison of uncertainty criteria for active SLAM," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2012, pp. 2080–2087.
[14] M. L. Rodríguez-Arévalo, J. Neira, and J. A. Castellanos, "On the importance of uncertainty representation in active SLAM," *IEEE Trans. Robot.*, vol. 34, no. 3, pp. 829–834, Jun. 2018.
[15] L. V. Montiel and J. E. Bickel, "Exploring the accuracy of joint-distribution approximations given partial information," *Eng. Economist*, vol. 64, no. 4, pp. 323–345, Oct. 2019.
[16] D. Trivun, E. Salaka, D. Osmankovic, J. Velagic, and N. Osmic, "Active SLAM-based algorithm for autonomous exploration with mobile robot," in *Proc. IEEE Int. Conf. Ind. Technol. (ICIT)*, Mar. 2015, pp. 74–79.
[17] L. Carlone, J. Du, M. K. Ng, B. Bona, and M. Indri, "Active SLAM and exploration with particle filters using Kullback–Leibler divergence," *J. Intell. Robotic Syst.*, vol. 75, no. 2, pp. 291–311, Aug. 2014.
[18] A. A. Ravankar, A. Ravankar, Y. Kobayashi, and T. Emaru, "Autonomous mapping and exploration with unmanned aerial vehicles using low cost sensors," *Multidisciplinary Digit. Publishing Inst. Proc.*, vol. 4, no. 1, p. 44, 2018.
[19] A. Dai, S. Papatheodorou, N. Funk, D. Tzoumanikas, and S. Leutenegger, "Fast frontier-based information-driven autonomous exploration with an MAV," 2020, *arXiv:2002.04440*.
[20] R. Shrestha, F.-P. Tian, W. Feng, P. Tan, and R. Vaughan, "Learned map prediction for enhanced mobile robot exploration," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, May 2019, pp. 1197–1204.
[21] S. Mai, "Simultaneous localisation and optimisation—Towards swarm intelligence in robots," Ph.D. thesis, Dept. Inform., Otto von Guericke Univ. Magdeburg, Magdeburg, Germany, 2018.
[22] F. Vanegas, K. J. Gaston, J. Roberts, and F. Gonzalez, "A framework for UAV navigation and exploration in GPS-denied environments," in *Proc. IEEE Aerosp. Conf.*, Mar. 2019, pp. 1–6.
[23] P. G. C. N. Senarathne and D. Wang, "Towards autonomous 3D exploration using surface frontiers," in *Proc. IEEE Int. Symp. Saf., Secur., Rescue Robot. (SSRR)*, Oct. 2016, pp. 34–41.
[24] C. Dornhege and A. Kleiner, "A frontier-void-based approach for autonomous exploration in 3D," *Adv. Robot.*, vol. 27, no. 6, pp. 459–468, 2013.
[25] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," *Res. Rep.*, vol. 98, no. 11, Oct. 1998.
[26] A. Bircher, M. Kamel, K. Alexis, H. Oleynikova, and R. Siegwart, "Receding horizon path planning for 3D exploration and surface inspection," *Auto. Robots*, vol. 42, no. 2, pp. 291–306, Feb. 2018.
[27] H. Umari and S. Mukhopadhyay, "Autonomous robotic exploration based on multiple rapidly-exploring randomized trees," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2017, pp. 1396–1402.
[28] C. Papachristos, S. Khattak, and K. Alexis, "Uncertainty-aware receding horizon exploration and mapping using aerial robots," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Jun. 2017, pp. 4568–4575.
[29] C. Papachristos, M. Kamel, M. Popovic, S. Khattak, A. Bircher, H. Oleynikova, T. Dang, F. Mascarich, K. Alexis, and R. Siegwart, "Autonomous exploration and inspection path planning for aerial robots using the robot operating system," in *Robot Operating System (ROS)*. Berlin, Germany: Springer, 2019, pp. 67–111.
[30] M. Faria, I. Maza, and A. Viguria, "Applying frontier cells based exploration and lazy Theta* path planning over single grid-based world representation for autonomous inspection of large 3D structures with an UAS," *J. Intell. Robotic Syst.*, vol. 93, nos. 1–2, pp. 113–133, Feb. 2019.
[31] S. Suresh, P. Sodhi, J. G. Mangelson, D. Wettergreen, and M. Kaess, "Active SLAM using 3D submap saliency for underwater volumetric exploration," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Aug. 2020, pp. 3132–3138.
[32] H. Lehner, M. J. Schuster, T. Bodenmüller, and S. Kriegel, "Exploration with active loop closing: A trade-off between exploration efficiency and map quality," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2017, pp. 6191–6198.
[33] R. Valencia and J. Andrade-Cetto, "Active pose SLAM," *Mapping, Planning and Exploration with Pose SLAM*. Berlin, Germany: Springer, 2018, pp. 89–108.

[34] S. A. Sadat, K. Chutskoff, D. Jungic, J. Wawerla, and R. Vaughan, "Feature-rich path planning for robust navigation of MAVs with mono-SLAM," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Jun. 2014, pp. 3870–3875.
[35] V. A. Rosas-Cervantes, Q.-D. Hoang, S.-G. Lee, and J.-H. Choi, "Multi-robot 2.5D localization and mapping using a Monte Carlo algorithm on a multi-level surface," *Sensors*, vol. 21, no. 13, p. 4588, Jul. 2021.
[36] Y. Kompis, L. Bartolomei, R. Mascaro, L. Teixeira, and M. Chli, "Informed sampling exploration path planner for 3D reconstruction of large scenes," *IEEE Robot. Autom. Lett.*, vol. 6, no. 4, pp. 7893–7900, Oct. 2021.
[37] A. J. Davison and D. W. Murray, "Mobile robot localisation using active vision," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 1998, pp. 809–825.
[38] A. J. Davison and D. W. Murray, "Simultaneous localization and map-building using active vision," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 7, pp. 865–880, Jul. 2002.
[39] E. Marchand and F. Chaumette, "Active vision for complete scene reconstruction and exploration," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 21, no. 1, pp. 65–72, Jan. 1999.
[40] S. Isler, R. Sabzevari, J. Delmerico, and D. Scaramuzza, "An information gain formulation for active volumetric 3D reconstruction," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2016, pp. 3477–3484.
[41] J. Delmerico, S. Isler, R. Sabzevari, and D. Scaramuzza, "A comparison of volumetric information gain metrics for active 3D object reconstruction," *Auto. Robots*, vol. 42, no. 2, pp. 197–208, 2018.
[42] S. Kriegel, C. Rink, T. Bodenmüller, and M. Suppa, "Efficient next-best-scan planning for autonomous 3D surface reconstruction of unknown objects," *J. Real-Time Image Process.*, vol. 10, no. 4, pp. 611–631, Dec. 2015.
[43] J. I. Vasquez-Gomez, L. E. Sucar, R. Murrieta-Cid, and E. Lopez-Damian, "Volumetric next-best-view planning for 3D object reconstruction with positioning error," *Int. J. Adv. Robotic Syst.*, vol. 11, no. 10, p. 159, Oct. 2014.
[44] J. A. Gibbs, M. P. Pound, A. P. French, D. M. Wells, E. H. Murchie, and T. P. Pridmore, "Active vision and surface reconstruction for 3D plant shoot modelling," *IEEE/ACM Trans. Comput. Biol. Bioinf.*, vol. 17, no. 6, pp. 1907–1917, Dec. 2020.
[45] T.-X. Zheng, S. Huang, Y.-F. Li, and M.-C. Feng, "Key techniques for vision based 3D reconstruction: A review," *Acta Automatica Sinica*, vol. 46, no. 4, pp. 631–652, 2020.
[46] R. Zeng, Y. Wen, W. Zhao, and Y.-J. Liu, "View planning in robot active vision: A survey of systems, algorithms, and applications," *Comput. Vis. Media*, vol. 6, no. 3, pp. 225–245, 2020.
[47] D. Meagher, "Geometric modeling using octree encoding," *Comput. Graph. Image Process.*, vol. 19, no. 2, pp. 129–147, Jun. 1982.
[48] X. Yang and S. Fang, "Effect of field of view on the accuracy of camera calibration," *Optik*, vol. 125, no. 2, pp. 844–849, Jan. 2014.
[49] J. Buquet, J. Zhang, P. Roulet, S. Thibault, and J.-F. Lalonde, "Evaluating the impact of wide-angle lens distortion on learning-based depth estimation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2021, pp. 3693–3701.
[50] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: An open-source robot operating system," in *Proc. ICRA Workshop Open Source Softw.*, vol. 3, Kobe, Japan, 2009, p. 5.
[51] P. Li, Z. Liu, and D. Huang, "3D map merging based on overlapping region," in *Proc. 3rd Int. Conf. Electron. Inf. Technol. Comput. Eng. (EITCE)*, Oct. 2019, pp. 1133–1137.
[52] S. Nobili, R. Scona, M. Caravagna, and M. Fallon, "Overlap-based ICP tuning for robust localization of a humanoid robot," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Jun. 2017, pp. 4721–4728.
[53] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Auton. Robot.*, vol. 34, no. 3, pp. 189–206, 2013.
[54] S. Bai, J. Wang, F. Chen, and B. Englot, "Information-theoretic exploration with Bayesian optimization," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2016, pp. 1816–1822.
[55] S. Bai, J. Wang, K. Doherty, and B. Englot, "Inference-enabled information-theoretic exploration of continuous action spaces," in *Robotics Research*. Berlin, Germany: Springer, 2018, pp. 419–433.
[56] G. Grisetti, C. Stachniss, and W. Burgard, "Improved techniques for grid mapping with rao-blackwellized particle filters," *IEEE Trans. Robot.*, vol. 23, no. 1, pp. 34–46, Feb. 2007.

IKER LLUVIA received the B.Sc. degree in computer science and the M.Sc. degree in computational engineering and intelligent systems from the University of the Basque Country, in 2016 and 2017, respectively. He is currently pursuing the Ph.D. degree in robust navigation of autonomous robots using vision techniques with the Technological Centre, Tekniker. He has been a Researcher with the Technological Centre, Tekniker, since 2016. He has participated in several industrial and research projects, both national and international, and he is also involved in projects that deal with mobile robotics, computer vision, and machine learning. As a part of the thesis, he has collaborated with the Autonomous Intelligent Systems Group, Albert Ludwig University of Freiburg, Germany, as a Guest Researcher in 3D active mapping and exploration. His research interests include robotics, autonomous systems, human–robot collaboration, and machine learning.

ELENA LAZKANO received the B.Sc. degree in computer sciences from the University of the Basque Country (UPV/EHU), in 1992, the M.Sc. degree in artificial intelligence from Katholieke Universiteit Leuven, Belgium, in 1993, and the Ph.D. degree in computer sciences from UPV/EHU, in 2004. She is currently an Associate Professor with the Computer Sciences and Artificial Intelligence Department, UPV/EHU. She is also the Co-Head of the Robotics and Autonomous Systems Group, Donostia-San Sebastian. She is also a Researcher in the field of intelligent robotics, being a member of the program committee of international conferences and a reviewer of international journals.

ANDER ANSUATEGI received the M.Sc. and Ph.D. degrees in computer science from the University of Basque Country (UPV/EHU), in 2005 and 2012, respectively. In October 2006, he was a Researcher at Tekniker. With more than ten years of experience in robotics research, he is currently the Coordinator of the research activities in robotics at Tekniker. During these ten years, he has taken part in numerous European projects. He has taken part in European projects, such as MAINBOT, ROBO-PARTNER, FOURBYTHREE, and GREENPATROL, which are related with sensors data management and robotics. He is coordinating the experiment PRODETECT running as part of the second open call of the COVR H2020 Project. He has some articles in peer-reviewed international journals and contributions to several congresses related to his field of expertise. His research interests include artificial intelligence, data management, planning and navigation in mobile robot, and machine learning.