Research Article

INTERNATIONAL JOURNAL OF ADVANCED ROBOTIC SYSTEMS

International Journal of Advanced Robotic Systems
September-October 2017: 1–15

DOI: 10.1177/1729881417732757
journals.sagepub.com/home/arx
SAGE

Multi-sensor three-dimensional Monte Carlo localization for long-term aerial robot navigation

Francisco J Perez-Grau[1], Fernando Caballero[2], Antidio Viguria[1] and Anibal Ollero[2]

Abstract

This article presents an enhanced version of the Monte Carlo localization algorithm, commonly used for robot navigation in indoor environments, which is suitable for aerial robots moving in a three-dimentional environment and makes use of a combination of measurements from an Red,Green,Blue-Depth (RGB-D) sensor, distances to several radio-tags placed in the environment, and an inertial measurement unit. The approach is demonstrated with an unmanned aerial vehicle flying for 10 min indoors and validated with a very precise motion tracking system. The approach has been implemented using the robot operating system framework and works smoothly on a regular i7 computer, leaving plenty of computational capacity for other navigation tasks such as motion planning or control.

Keywords

Aerial robotics, localization, autonomous vehicle navigation

Date received: 17 February 2017; accepted: 29 July 2017

Topic: Mobile Robots and Multi-Robot Systems
Topic Editor: Nak-Young Chong
Associate Editor: Euntai Kim

Introduction

The new manufacturing paradigm pursues the acceleration of delivery rates through the gradual implementation of automated processes. Aerial robots can contribute to this automation process by providing some intrinsic capabilities such as flexibility, fast response, and availability. Figure 1 shows an aircraft manufacturing plant where small aerial robots are expected to begin carrying out specific logistic operations. In order to allow the safe introduction of aerial robots collaborating with humans in manufacturing plants, there is a need for robustness and reliability in a series of key enabling technologies such as localization, mapping, and path planning. Among them, the localization problem is essential for building a mobile robotic system, since accurate pose estimation is required for even the most basic tasks, such as holding the position. It is commonly referred to as "the most fundamental problem to providing a mobile robot with autonomous capabilities."[1] Once the aerial robot is capable of self-localizing with enough accuracy, the rest of technologies such as navigation and guidance can be implemented.

The localization problem consists in accurately determining the current pose of a robot (position and orientation) in a specific environment. This has been an active research

[1] Center for Advanced Aerospace Technologies (CATEC), La Rinconada, Sevilla, Spain
[2] Department of System Engineering and Automation, University of Seville, Seville, Spain

Corresponding author:
Francisco J. Perez-Grau, Center for Advanced Aerospace Technologies (CATEC), Aerospace Technology Park of Andalusia, 41309 La Rinconada, Sevilla, Spain.
Email: fjperez@catec.aero

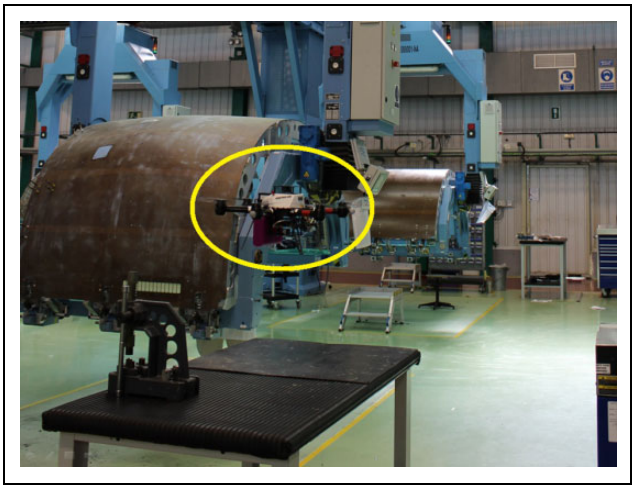

Figure 1. Aerial robot taking off from a designated location to perform logistic operations in an aircraft manufacturing plant.

topic for the past decades. Ground robots moving in 2-D environments have demonstrated good performances in terms of localization either indoor[2] or on streets and highways.[3,4] However, aerial robots operate and move in 3-D environments and are exposed to additional challenges apart from the space dimensionality, since onboard capabilities are limited due to payload, computational resources, and power constraints.

Outdoor operation for aerial robots based on global positioning system (GPS) can be generally assumed due to the existence of low-cost commercial products offering mature performances in a wide variety of applications.[5–7] Despite this, autonomous aerial vehicles are starting to play a major role in applications such as inspection,[8] search and rescue,[9] or security surveillance,[10] which usually require the aerial robots to fly in dense environments, at low altitudes or indoors. In such cases, GPS signals are often shadowed or can be unavailable. In addition, the update frequency of GPS location queries is not enough to achieve the aforementioned tasks in a robust and safe manner, since the aerial robots are supposed to be operating in small environments and move at relatively high speeds.

Related work

In order to compensate the issues that GPS poses, other methods must be used to provide accurate and robust robot localization. Vision-based approaches for robot localization and odometry are very popular due to the affordability and availability of cameras and the low weight of the sensors. However, one important drawback is the associated processing complexity. It is very common to find approaches that make use of monocular cameras fused with inertial sensors in order to perform visual simultaneous localization and mapping (SLAM).[11–14] However, these approaches tend to fail when the aerial vehicle exhibits high-speed motions. Other approaches make use of photogrammetry[15] but pose additional processing requirements that hinder online computation.

Another trend is to employ stereovision,[16–18] which allows for calculation of 3-D depth measurements at a higher computational cost. More recently, Red,Green,Blue-Depth (RGB-D) sensors based on structured light have recently become a very popular option[19,20] due to their low weight, low cost, and the amount of information provided. Besides RGB images, they directly provide depth maps of the scene in front of the sensor, saving the burden of 3-D reconstruction computation. Besides, this sensor exhibits another important advantage with respect to classic camera-based approaches: Depth estimation does not depend on the availability of distinct visual features in the scene and is less reliant on lighting conditions while operating indoors. These factors make RGB-D sensors more suitable for reliable localization in our framework. There are several state-of-the-art algorithms available that provide localization estimations, such as RGBD-SLAM,[19] CCNY RGB-D,[21] or RTAB-Map.[22] They exhibit good accuracies under realistic conditions; however, when applied to aerial robots for online localization estimation, they drop their performance due to vibrations and faster motions than ground robots, as shown in the section "Experimental results." They may pose safety issues when performing online localization in order to close the control loop, since the map building process that they perform at the same time may lead to localization jumps in order to fit the current sensor point cloud to the map.

In general, vision-based odometry and localization systems for aerial robots are not reliable enough in the long term due not only to cumulative drift but also to external factors such as poor illumination, lack of texture, occlusions, or moving objects. All these have a significant impact on the robustness and the reliability of the most state-of-the-art algorithms. The aforementioned approaches demonstrate fairly good results in the short term; however, they could quickly diverge depending on the environment. Some of them perform well even in long trajectories when we revisit the same areas, which allows for loop closing in order to reduce the localization uncertainty.[23,24] However, the problem of loop closing in order to distinguish places in the environment is usually framed as a classification task rather than a robot localization task. Nevertheless, reliable place recognition can be challenging in large-scale structured environments that might exhibit similar scenes in different areas, such as a manufacturing plant.

Other approaches that cope with long-term localization solutions based on vision rely on previously built maps of the environment. Logistic services are usually carried out in known environments, and this is also a requirement for path planning because human workers should be able to specify goals to the aerial robots in a predefined coordinate system or a map. In addition, map-based approaches for localization are usually better in terms of reliability and computational requirements. An a priori map can be computed off-line using cameras and/or range sensors, making use of existing algorithms, for example, RGBD-SLAM.[25]

One of the approaches based on a predefined model of the environment is commonly known as teach-replay,[26,27] which is accomplished in two stages: First the robot is manually piloted along the desired path as in a teaching phase, and an accurate 3-D map of the environment is built, along with the robot motion from this learning path; afterward, this map is used to locate the robot when it repeatedly visits the same path. However, this approach is somewhat simplistic and limited since it only enables a robot to follow a predetermined trajectory.

Monte Carlo localization (MCL) is another approach that makes use of a known map of the environment and is commonly used for robot navigation in indoor environments.[28] It is a probabilistic localization algorithm that makes use of a particle filter to estimate the pose of the robot within the map, based on sensor measurements. Important benefits include the possibility of accommodating arbitrary sensor characteristics, motion dynamics, and noise distributions. There is a variant of MCL called adaptive MCL (AMCL). The term "adaptive" comes from the fact that the number of particles is adjusted dynamically: If there is high uncertainty about the robot pose, the number of particles increases; if the pose is well known, the number decreases. An open-source version of AMCL is included in the robot operating system (ROS) navigation stack (http://wiki.ros.org/amcl). However, it is meant for wheeled robots moving in a 2-D environment, requiring a 2-D laser scanner for map building and localization. Other authors presented an extension of this approach for 6-D localization based on 2-D laser scanner,[29] but it is meant for the 2-D motion of humanoid robots in a 3-D environment, which makes it not suitable for aerial robots. The use of GPUs can enhance the computing capabilities of the onboard system regarding MCL-based approaches, as proposed by other authors.[30,31]

Localization based on radio beacons has also been deeply studied in the last years, with specific setups taking place in manufacturing environments[32] and using aerial robots.[33,34] Radio ranging sensors provide point-to-point distance measurements and offer a low-cost solution that can be implemented in almost any scenario, with the advantage of not requiring a direct line of sight between each pair of sensors. Besides, the data association problem is trivially solved by attaching the sensor identification to the range measurement information. Ultra-wideband (UWB) is a wireless communication technology which has attracted interest from the research community as a promising solution for precise target localization and tracking.[35–37] It is particularly well suited for short-distance indoor applications, using several fixed sensors placed at known positions in the environment, and a mobile sensor onboard the aerial robot. Hence, the position estimation of the onboard sensor can be obtained by triangulation with an accuracy of the order of that from the sensors. However, these sensors are poorly suited to constitute a full localization system due to the lack of bearing information, thus leading to multiple location hypotheses. Combining error-bounded range sensing with reliable short-term position estimation based on visual odometry has been already studied for achieving robust long-term localization in position.[38] The aerial robot orientation was not robustly determined by this approach though; yaw angle was solely determined by the visual odometry algorithm, while roll and pitch angles were assumed to be observable with an inertial measurement unit (IMU).

The article is structured as follows. The section "System overview" briefly describes the overall approach. In the section "Localization approach," the main algorithm used in this work is detailed, an enhanced MCL approach. Experiments demonstrating long-term localization are summarized in the section "Experimental results," followed by conclusions and future work.

System overview

The aforementioned approaches (visual odometry, MCL, and radio-based localization) are promising in that they can all provide solutions to the localization problem, but important drawbacks are present using each approach alone. The main contribution of this work focuses on the combination of all three: an MCL algorithm relying on a previously built 3-D map, an RGB-D sensor for odometry and point cloud matching, and radio-based sensors installed in the environment and localized within the map. This results in a system suitable for long-term aerial robot localization, in a way that these technologies benefit from each other. We are able to have a reliable short-term position estimation based on visual odometry, while keeping its drift bounded thanks to the map matching and the integration of range measurements. The noise and outliers present in range sensing are filtered thanks to the odometry prior. Finally, the MCL exploits the odometry to update the motion of the particles and the range measurements to maintain a low dispersion on the position of particles whenever the point cloud matching with the map is not acceptable. This solution offers reliable localization in position and yaw angle, while roll and pitch angles can still be observable through an IMU. Moreover, the implemented algorithms are highly efficient so they are suitable for real-time operation onboard the aerial robot for performing online localization. The use of GPUs is discarded; we propose an approach that can be implemented in standard CPUs and hence not limited its applicability in regular platforms.

The block diagram in Figure 2 depicts an overview of our method. The aerial robot localization solution is based on the integration of visual odometry, 3-D point clouds, and distance measurements to known radio beacons into an MCL.

The odometry system combines images, point clouds, pose estimation, and key-framing in a loose-coupling filter in order to estimate a reliable and accurate localization at the short term. As it was previously mentioned, odometry

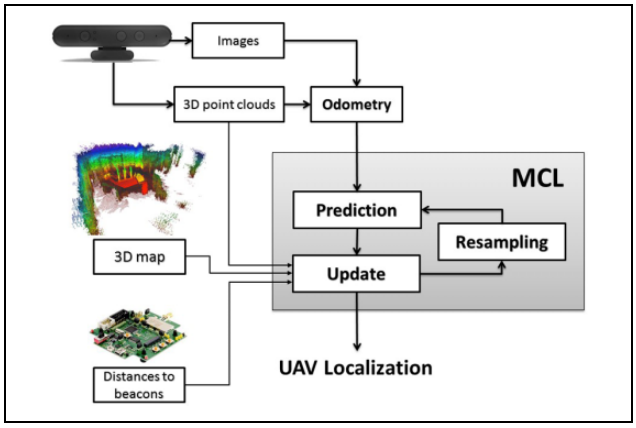

Figure 2. Schematic overview of the processing pipeline.

will accumulate errors over time. MCL takes care of detecting and correcting the cumulative errors of the odometry by comparing the point clouds with a known 3-D map of the area and using range sensing to fixed beacons to readjust localization errors.

This article shows how the integration of all the elements and sensing approaches was carried out in order to get a fast, reliable, and error-bounded localization.

Localization approach

Robot localization involves the creation of a map of the environment, which typically is based on some sort of SLAM approach. This would later allow the estimation of the robot pose (position and orientation) in the coordinate frame of such map. Most localization algorithms require an accurate odometry in order to maintain a reliable robot pose estimation in the map. However, odometry is subject to drift as a result of error accumulation over time, and there is a need for some other source of information in order to reset such drift.

Our work extends the MCL proposed by Hornung et al.[29] As previously stated, this algorithm needs a map of the environment in order to estimate the robot pose within the map based on its motion and sensing. The starting point of the algorithm is an initial belief of the robot pose probability distribution, which determines the distribution of the particles around such pose according to such belief. These particles are then propagated following the robot motion model each time its pose changes. Every time we receive new sensor readings, each particle evaluates its accuracy by checking how likely it would receive such sensor reading at its current pose. The next step of the algorithm is redistributing (resampling) the particles to new poses that are more likely to be accurate. This is an iterative process that involves moving, sensing, and resampling, while all the particles should converge to a single cluster near the true pose of the robot.

The motion model used in this work is based on a fast and reliable visual odometry computed from RGB-D data. We have adapted a stereovision algorithm[38] (which is publicly available at http://wiki.ros.org/viodom) in order to make it work with RGB-D sensors. The odometry estimation is applied to the particles, providing an estimation of the a priori distribution of the MCL. The sensor readings are the point clouds provided by the RGB-D sensor and the distance measurements to several radio-tags installed in the environment. The weight of each particle is then calculated according to the two types of sensor readings. The point clouds are transformed to each particle pose in order to find correspondences between the cloud and what the map should look like from that particle's pose. The distance measurements to radio-tags are used to check how well each particle position matches such distances. In order to fuse both types of sensor readings, each particle has a weight for each type of sensor and they are later fused into a single weight.

The particle filter consists of N particles $\mathbf{p}_i$, each one with the following state vector

$$\mathbf{p}_i = \begin{bmatrix} x \\ y \\ z \\ \psi \end{bmatrix} \tag{1}$$

where ψ refers to the yaw angle of the aerial robot. Note how the robot roll and pitch angles are not included into the robot pose definition. We assume that these angles are available and accurate enough in an unmanned aerial vehicle (UAV) through the use of its onboard IMU. They are fully observable and their values are usually accurate in aerial robots because they are the most basic control variables (together with the rotation rates) for the system stability.[39] While it is true that roll and pitch estimation based on accelerometer and gyroscope integrations might be biased under constant accelerations (e.g. loitering in fixed-wing UAVs), these scenarios are very difficult to achieve indoors and hence are not considered in our approach. Besides, this greatly reduces the computational complexity of the algorithm, allowing for real-time onboard computation.

Each particle has an associated weight w_i such as

$$\sum_{i=1}^{N} w_i = 1 \tag{2}$$

Filter initialization

Particles can be initialized manually or automatically by setting the initial position together with a covariance matrix to distribute the particles in the space.

In the case of manual initialization, the particle cloud is sampled following a 4-D normal distribution using as mean the provided initial position and a specific covariance matrix Σ. The associated weights w_i are initialized to $1/N$, uniformly.

Automatic initialization can handle the "kidnapped robot" problem. Particles are drawn uniformly over the

4-D state (x, y, z, and ψ) into the whole map. As soon as the UAV starts moving, the filter updates and good hypotheses start gaining weight, while low-weight particles are shifted toward more interesting areas in the resampling. The section "Experimental results" includes an example of automatic initialization in order to demonstrate the capabilities of our approach.

Nevertheless, for achieving full autonomous operation of an aerial robot before takeoff, we must manually initialize the starting pose of the robot, providing an initial position and orientation along with a covariance matrix to distribute the particles around such initial pose. Knowing the starting takeoff location of an aerial robot will usually be the case when deploying an autonomous system in real scenarios, for example, in a manufacturing plant for logistic operations.

Besides, the risk of getting stuck in highly symmetric environments is smaller when combining the 3-D point clouds with radio range measurements. Particles that do not fit distance measurements to fixed beacons will be rapidly discarded.

Filter prediction

The odometry system periodically provides increments for each of the components of the robot state vector

$$\Delta \mathbf{p}_i = \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta z \\ \Delta \psi \end{bmatrix} \tag{3}$$

This information is used to compute the a priori distribution of the particles. Thus, considering that the multirotors are holonomic systems, the state of the particles will evolve according to the following expressions

$$x_i^{t+1} = x_i^t + \Delta x * \cos(\psi_i^t) - \Delta y * \sin(\psi_i^t) \tag{4}$$

$$y_i^{t+1} = y_i^t + \Delta x * \sin(\psi_i^t) + \Delta y * \cos(\psi_i^t) \tag{5}$$

$$z_i^{t+1} = z_i^t + \Delta z \tag{6}$$

$$\psi_i^{t+1} = \psi_i^t + \Delta \psi \tag{7}$$

The values of Δx, Δy, Δz, and $\Delta \psi$ are drawn randomly following a normal distribution centered in their actual values and standard deviations proportional to each increment itself, for example $\sigma_x = k_x * \Delta x$ with $k_x > 0$. The value of k_x is always positive, and it is a design parameter that depends on the accuracy of the odometry system.

Filter update

We have defined a threshold in both position and orientation such that if the visual odometry exceeds any of those, a filter update is performed using the last 3-D point cloud received from the RGB-D camera, and all the distance measurements to the UWB beacons are received since the last filter update. Then, each particle $\mathbf{p}_i$ evaluates its relative importance by checking how likely it would receive such sensor reading at its current pose, computing a new weight value w_i. This is performed through the use of a 3-D occupancy grid of the environment in the form of an OctoMap[40] and augmented with the position of the fixed locations of radio beacons.

Given the distinct nature of the two technologies involved, we calculate separate weights for each sensing modality. A weighted average is used to obtain the final weight of each particle

$$w_i = \alpha * w_i^{\text{map}} + (1 - \alpha) * w_i^{\text{range}} \tag{8}$$

where α is chosen depending on the particularities of the indoor environment where the UAV is going to operate. If the map used in the MCL does not contain the full environment or its accuracy is not enough to trust the map matching, α should be lower than 0.5. Whereas if there are few UWB sensors deployed in the environment or their location is not accurate, α should be higher. The experimental results are obtained with different values of α in order to show its impact.

Computation of w_i^{map}: The acquired RGB-D point cloud is transformed to each particle pose in order to find correspondences between such cloud and what the map should look like from that particle's pose. Since this is very expensive computationally, we first compute a 3-D probability grid as in the study by Hornung et al.,[29] in which each position stores a value of how likely it is that such position falls within an occupied point of the map, instead of storing binary information about occupancy as in the provided map. Each 3-D position p_i of the grid is then filled with probability values according to a specific Gaussian distribution centered in the closest occupied point in the map from p_i, map_i, and whose variance σ^2 depends on the sensor noise used in the approach

$$\text{grid}(p_i) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-||p_i - \text{map}_i||^2 / 2\sigma^2} \tag{9}$$

Such probability grid only needs to be computed once, is not required to be updated for a given environment, and relieves from performing numerous distance computations between each cloud point for each particle and its closest occupied point in the map. Besides, each point cloud is first transformed according to the current roll and pitch provided by the onboard IMU. This transformation is done just once per update, reducing the computational requirements as well.

Then, for every point of the transformed cloud, we access its corresponding value in the 3-D probability grid. Such value would be an indicator of how likely is that point to be part of the map. By doing this with every point of the cloud and adding all the probability values, we obtain a

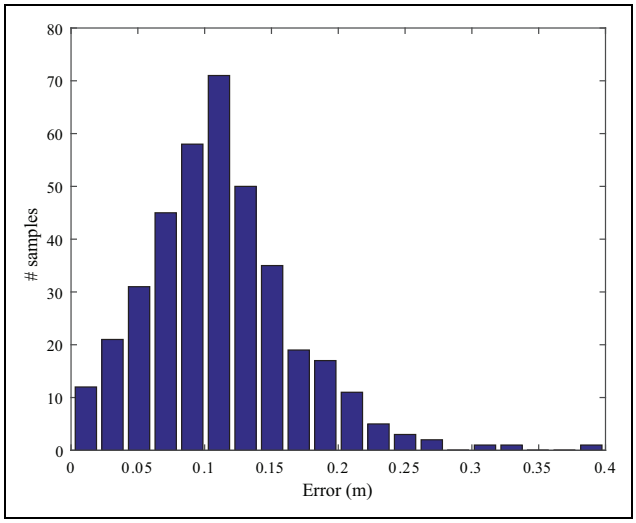

Figure 3. Error histogram characterization of the range sensor for indoor environments.

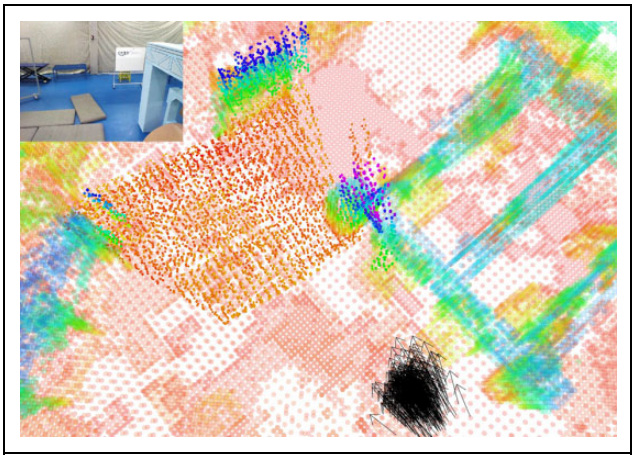

Figure 4. Particle cloud within the 3-D map (black arrows), and associated 3-D point cloud projected from the weighted sum of all particles along with the 3-D map of the environment. Top left corner shows current RGB image.

figure of how well that particle fits the true location of the aerial robot according to the map.

Finally, the weight w_i of each particle $\mathbf{p}_i$ is computed. Assuming that the point cloud is composed of M 3-D points $\mathbf{c}_j$, the weight is computed by adding all the associated probability grid values as follows

$$w_i^{\text{map}} = \frac{1}{M}\sum_{j=1}^{M} \text{grid}(\mathbf{p}_i(\mathbf{c}_j)) \tag{10}$$

where $\mathbf{p}_i(\mathbf{c}_j)$ stands for the transformation of the point to the particle's state and $\text{grid}(\mathbf{p}_i(\mathbf{c}_j))$ is the evaluation of the probability grid in such transformed position.

Computation of w_i^{range}: On the other hand, a set of fixed radio-based sensors is used to localize the aerial robot, which carries another radio-based sensor. They provide distance measurements between them which could be used to constrain localization errors produced by both the odometry and the matching between the point clouds and the map. The sensors used in our approach are low-cost UWB that can be easily found in the market. The distance measurements have a standard deviation of roughly 0.1 m after filtering potential outliers. Figure 3 shows a histogram of the measurement errors for indoor operation, generated after collecting nearly 400 samples. It can be seen how the measurements follow a Gaussian distribution with some outliers.

The distance measurements are integrated into the particle filter as follows. Since the radio beacons do not provide bearing information, we first define new particle states without ψ (yaw angle)

$$\mathbf{p}_i^{\text{range}} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{11}$$

and sensor UWB beacon states using their known positions

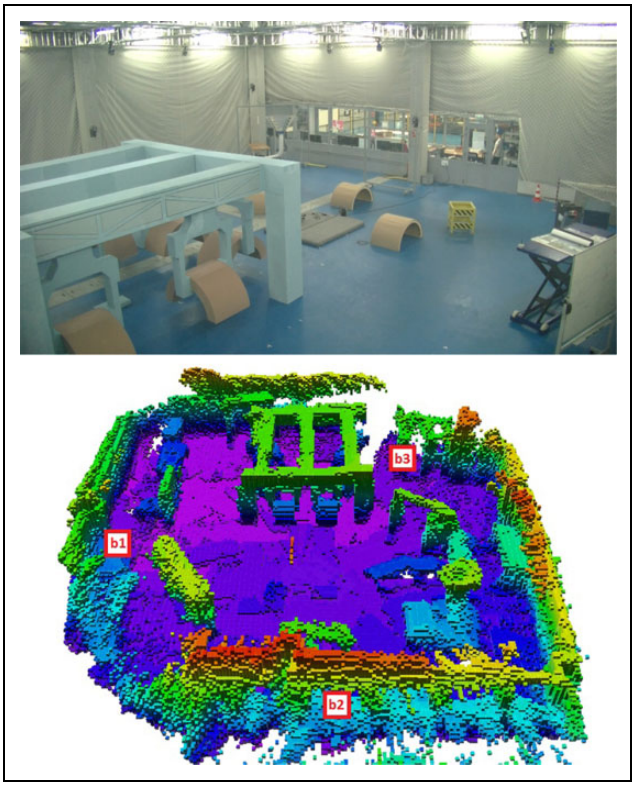

Figure 5. CATEC indoor test bed and scenario used for the field experiments. Picture of the site of the experiments (top). 3-D map of the area with approximate UWB beacon locations (b1, b2, and b3) (bottom). CATEC: Center for Advanced Aerospace Technologies; UWB: ultra-wideband.

$$\mathbf{b}_k = \begin{bmatrix} x_k \\ y_k \\ z_k \end{bmatrix} \tag{12}$$

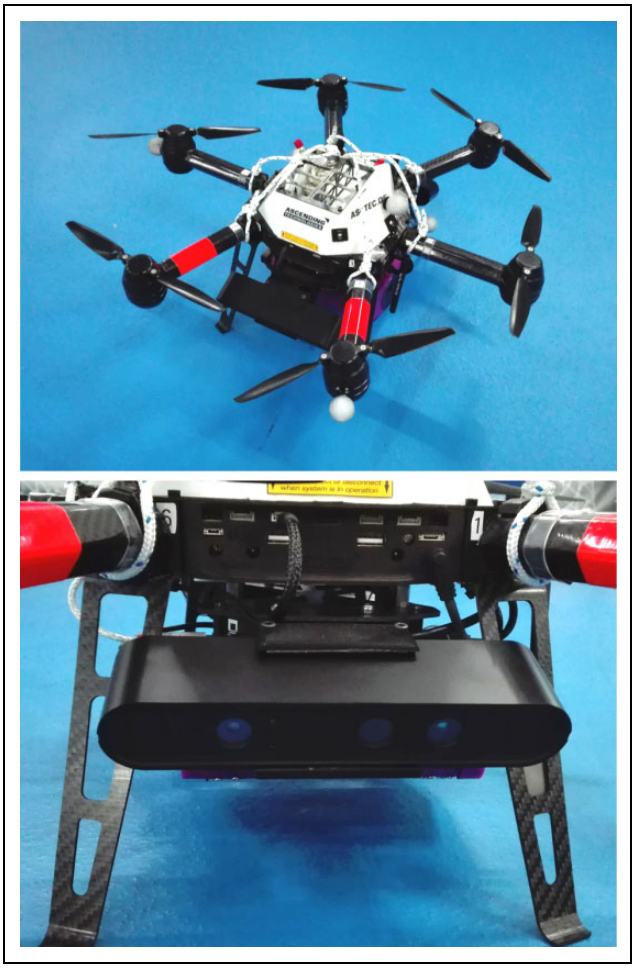

Figure 6. Aerial robot with RGB-D sensor at the front.

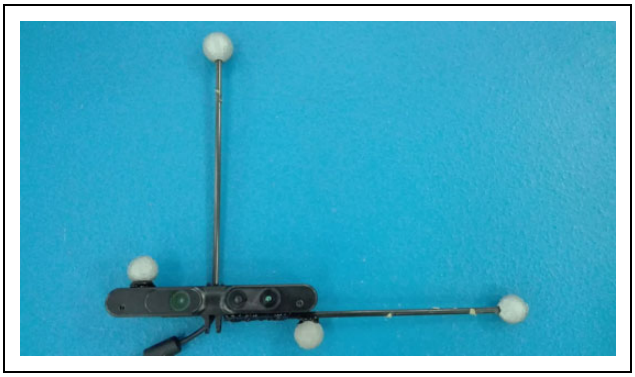

Figure 7. RGB-D sensor with passive markers for precise 3-D map building.

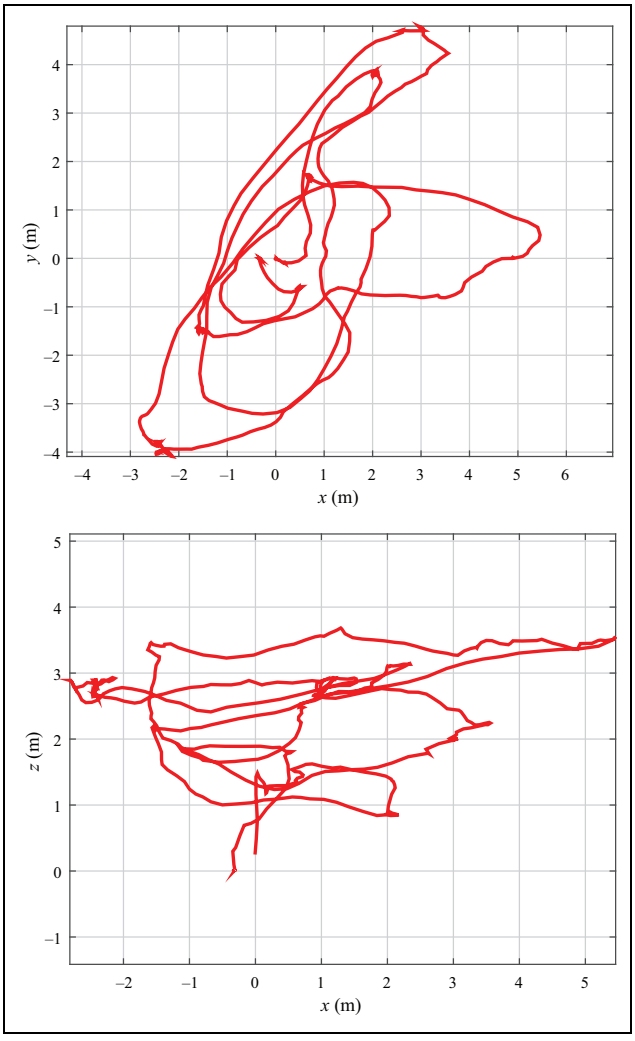

Figure 8. Ground-truth trajectory followed by the aerial robot during the experiment. *X*–*Y* trajectory (top). *X*–*Z* trajectory (bottom).

Given the measured radio-based distance r_k from the aerial robot to the k th beacon, the following constraint can be applied to each particle state

$$r_k = ||\mathbf{p}_i^{\text{range}} - \mathbf{b}_k|| \tag{13}$$

This constraint can be easily applied to the particle weight calculation according to the computed distance d_{ik} between the i th particle $\mathbf{p}_i^{\text{range}}$ and the k th beacon $\mathbf{b}_k$, and how close this is to r_k. A Gaussian distribution with mean r_k and a standard deviation σ of roughly 0.1 m (which fits the error histogram in Figure 3) is used in order to obtain a probability value. To aggregate the values from the measurements of different beacons, the product is used because they are independent probabilistic processes. The weight of the i th particle associated with range sensing is calculated as follows

$$w_i^{\text{range}} = \prod_{k=1}^{L} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(r_k - d_{ik})^2/2\sigma^2} \tag{14}$$

where L is the number of beacons in the environment.

A great advantage of this approach is that the distance to a single sensor is enough to update the weights of the particles, it does not need to wait until distances to three or more sensors are obtained by the onboard radio-tag, which usually happens in other state-of-the-art range-based localization systems.

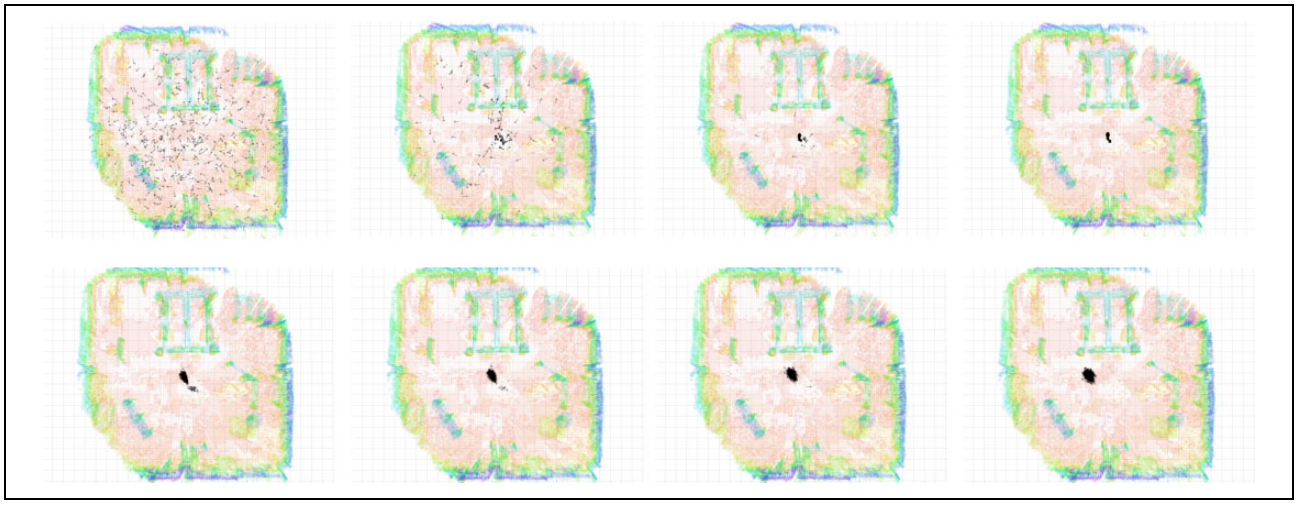

Figure 9. Automatic initialization of particles. From left to right and top to bottom, time evolution of particles after automatic initialization. Black arrows represent the particles.

Weight combination: Before combining the weights of both sensing approaches according to equation (8), all the weights must first be normalized within their categories in order to verify equation (2) and represent a valid probability distribution. This also prevents combining weights of distinct nature, which usually are in different orders of magnitude. Equation (8) is used to obtain the final weight of each particle, and the new weights are normalized again in order to represent a valid probability. The experimental results are first obtained with $\alpha = 0.5$, and then this parameter is modified in order to evaluate its impact.

The authors also evaluated the option of drawing new particles into the filter with each new range measurement taking into account the lack of bearing information. Thus, with each range measurement, a set of particles would be drawn in a sphere surrounding the onboard sensor position into the map. Most of these particles will be later on destroyed in the next resampling after update thanks to the weight contribution of the point cloud matching with the 3-D map. However, this approach was discarded because it requires a large number of particles for a proper representation of the range hypotheses, which hinders the computational efficiency.

Finally, the particle set is resampled to allow spreading the particles over the maximum likelihood areas. The algorithm employed for resampling is the low-variance sampler.[41]

The updated state vector for the aerial robot is then calculated as the weighted sum of all the particles. Due to this mixed approach, the outliers commonly present in indoor range measurements are rejected thanks to the particle weighting. If an outlier in the distance measurement from a range sensor is received, following equation (14), this would result in very low values for w_i^{range}; in this case, the resampling would only depend on the associated weights calculated from the point cloud matching, and the relative scoring among the set of particles would remain unaffected. Figure 4 shows the particle cloud within the map and the 3-D point cloud projected from the pose associated with the weighted sum of all the particles.

Table 1. MCL parameters.

Parameter	Value
Number of particles	500
α (equation (8))	0.5
OcTree resolution	0.1 m
RGB-D sensor σ	0.05 m
UWB sensor σ	0.1 m
Update threshold (pos)	0.1 m
Update threshold (yaw)	0.1 rad
k_x	0.4
k_y	0.4
k_z	0.2
k_ψ	0.5
Initial σ (pos)	0.2 m
Initial σ (yaw)	0.2 rad

MCL: Monte Carlo localization.

Experimental results

An experimental setup has been conceived to validate the presented approach. The field test took place at the indoor test bed of the Center for Advanced Aerospace Technologies (CATEC). This facility is used to develop and test a wide range of technologies related to autonomous systems. There is a useful volume of $15 \times 15 \times 5$ m^3; part of it can be seen in Figure 5. The test bed houses an indoor localization system based on 20 tracking cameras, which only needs the installation of passive markers on the objects to locate and/or track. This system is able to provide the

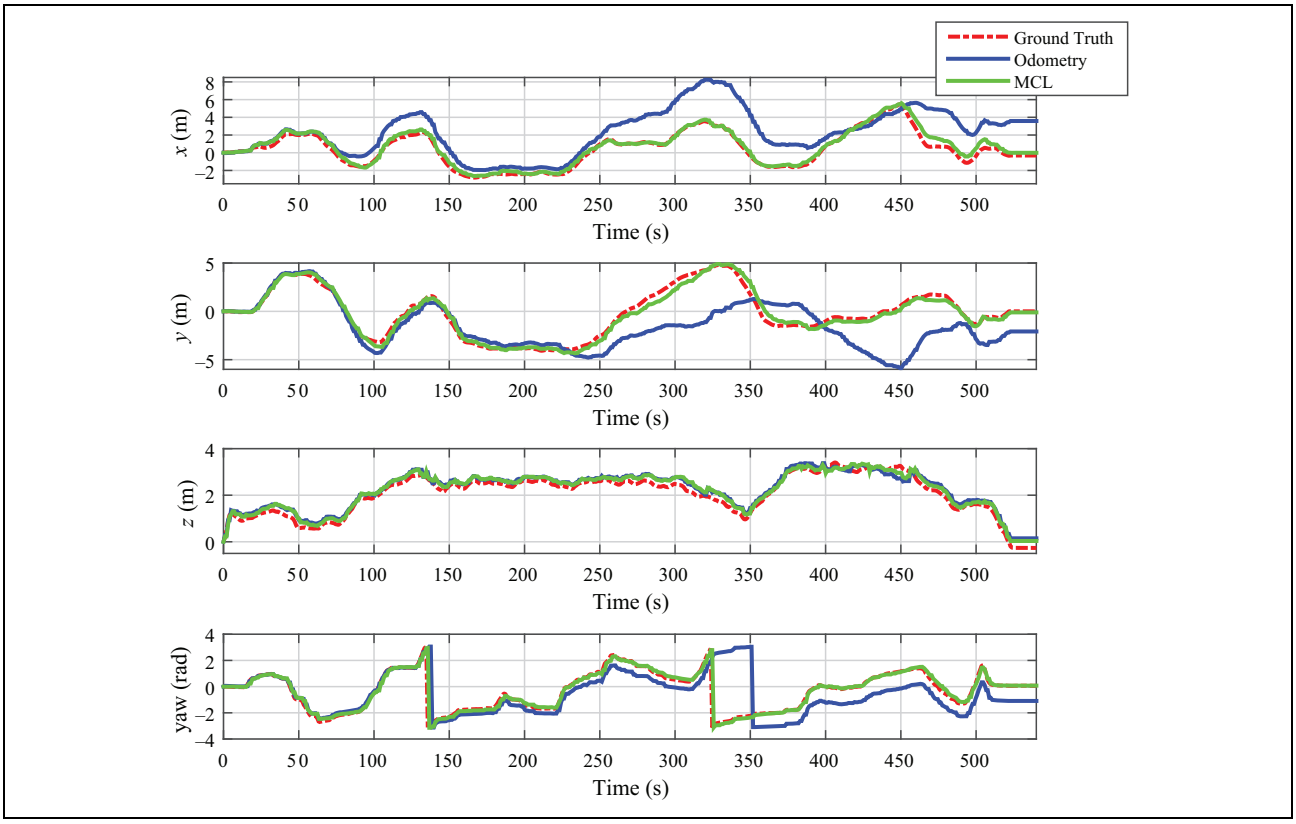

Figure 10. Localization results. UAV ground-truth position and yaw captured by the motion capture system (red line). Estimated UAV position and orientation using the proposed odometry (blue line). Estimated UAV position and orientation using the proposed approach (green line). UAV: unmanned aerial vehicle.

position and attitude of each object in real time with millimetric precision.

This section provides experimental results from the estimated localization compared with ground truth data obtained from the test bed tracking system. Prior to the discussion of the experimental results, the setup describing all the elements involved in the experiment is presented in the next subsection.

Experiment setup

The aerial robot in Figure 6 performed a flight based on VICON telemetry for localization and navigated using joystick commands that sent nearby position and orientation waypoints in the robot coordinate frame. The results presented in this article were obtained off-line. In order to validate the long-term character of our approach, the flight took roughly 9 min so the visual odometry could drift enough to verify that the two sensing measurements used in the particle filter help preventing localization error growth.

The complete experimental setup is composed of the following elements:

- An aerial robot with the following onboard sensors: an RGB-D sensor, an IMU, and a radio range sensor (see Figure 6).
- A 3-D map of the working area (see Figure 5) obtained using another RGB-D sensor, with a resolution of 0.1 m.
- Three radio range sensors installed across the indoor test bed at known positions.

The 3-D map of the area was acquired using another RGB-D sensor whose housing was augmented with several passive markers (see Figure 7) in order to get its pose with high accuracy from the test bed's motion capture system. Such sensor was carried around the test bed while connected to a laptop, and the acquired point clouds were projected using the current sensor pose and merged into an accurate octree. This map can be seen in Figure 5.

A small infrastructure of range sensors (three sensors) has been installed in the indoor scenario where the flight has taken place. The radio-based sensors have been installed at three different locations homogeneously distributed within the indoor test bed. These fixed locations were acquired using the VICON tracking system and provided to our filter. There is an additional node

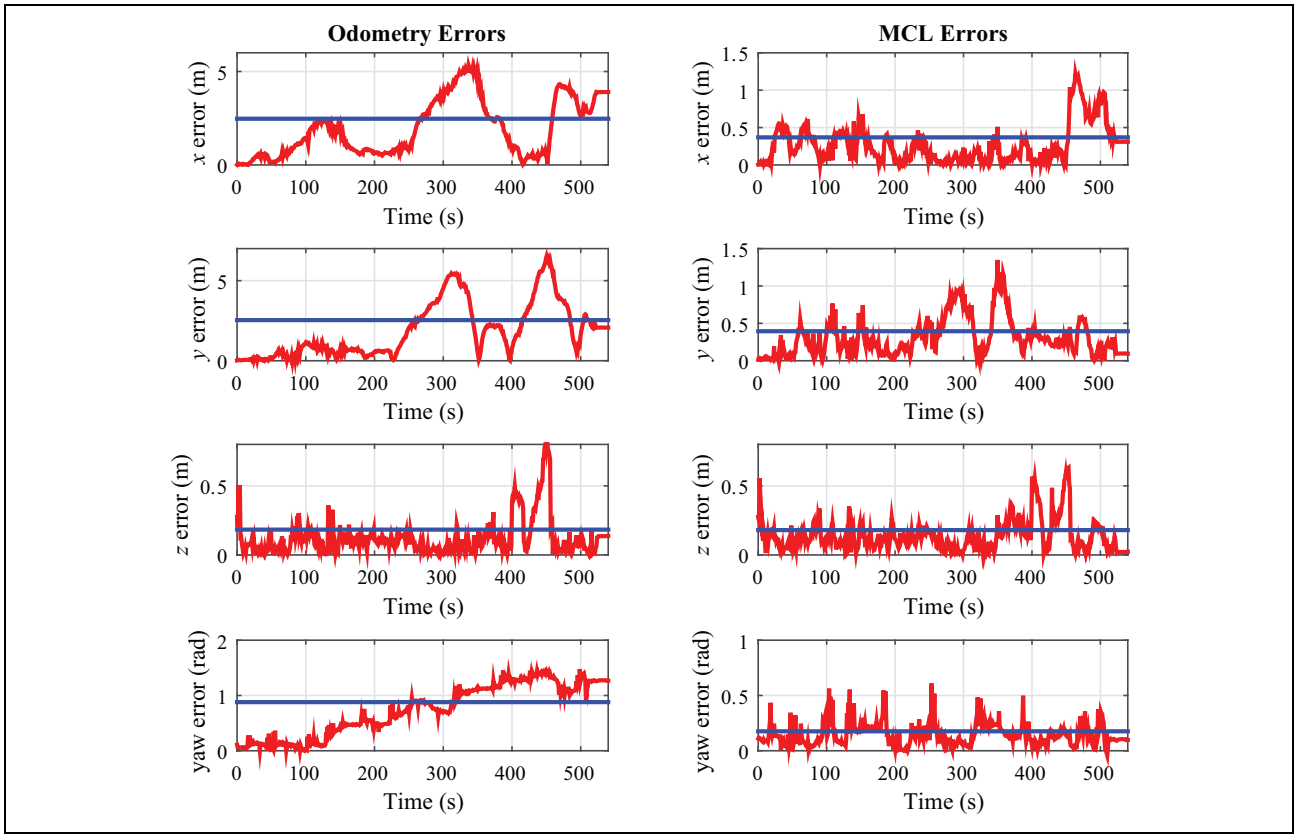

Figure 11. Position and orientation errors through time. Left plots correspond to the odometry approach, while right plots show results from the complete proposed approach including the MCL. Computed error of the estimated position and orientation with respect to the ground truth (red line). Estimated RMS error for each axis (blue line). RMS: root mean square; MCL: Monte Carlo localization.

installed on the aerial robot in order to compute point-to-point distances.

The aerial robot was remotely piloted through the test bed area at different heights. The ground-truth trajectory followed by the vehicle is presented in Figure 8. This is roughly 200-m long trajectory in which the robot performed a trajectory for a whole battery duration.

Automatic initialization

Even though for the experiment we manually set the initial pose of the aerial robot, an example of automatic initialization is shown in Figure 9. This demonstrates the capabilities of our approach toward handling the case in which the initial pose of the aerial robot is unknown. During takeoff, the filter starts performing prediction, update, and resampling of particles, based on the distance thresholds that were defined in Table 1. The result reveals how the UAV does not need to move much in order to converge a solution in position (top row in Figure 9). This mainly occurs because the radio ranging measurements quickly induce the right location. However, some of these particles do not represent the correct yaw angle. As soon as the UAV starts moving in the horizontal plane toward some obstacles, the point cloud matching with the map refines the position and properly approximates the yaw angle (bottom row in Figure 9).

Table 2. RMS localization errors.

	x (m)	*y* (m)	*z* (m)	yaw (rad)
Odometry	2.47	2.53	0.19	0.88
MCL	0.36	0.39	0.17	0.18

RMS: root mean square; MCL: Monte Carlo localization.

Results and discussion

The main objective of the experiment is to demonstrate the suitability of the approach in real time during regular operations. The estimated UAV position and yaw during the experiment can be seen in Figure 10, both from the visual odometry and the MCL. Roll and pitch angles are not included in the plots because, as previously stated, they are observable from the onboard IMU and are directly integrated into our localization approach. The

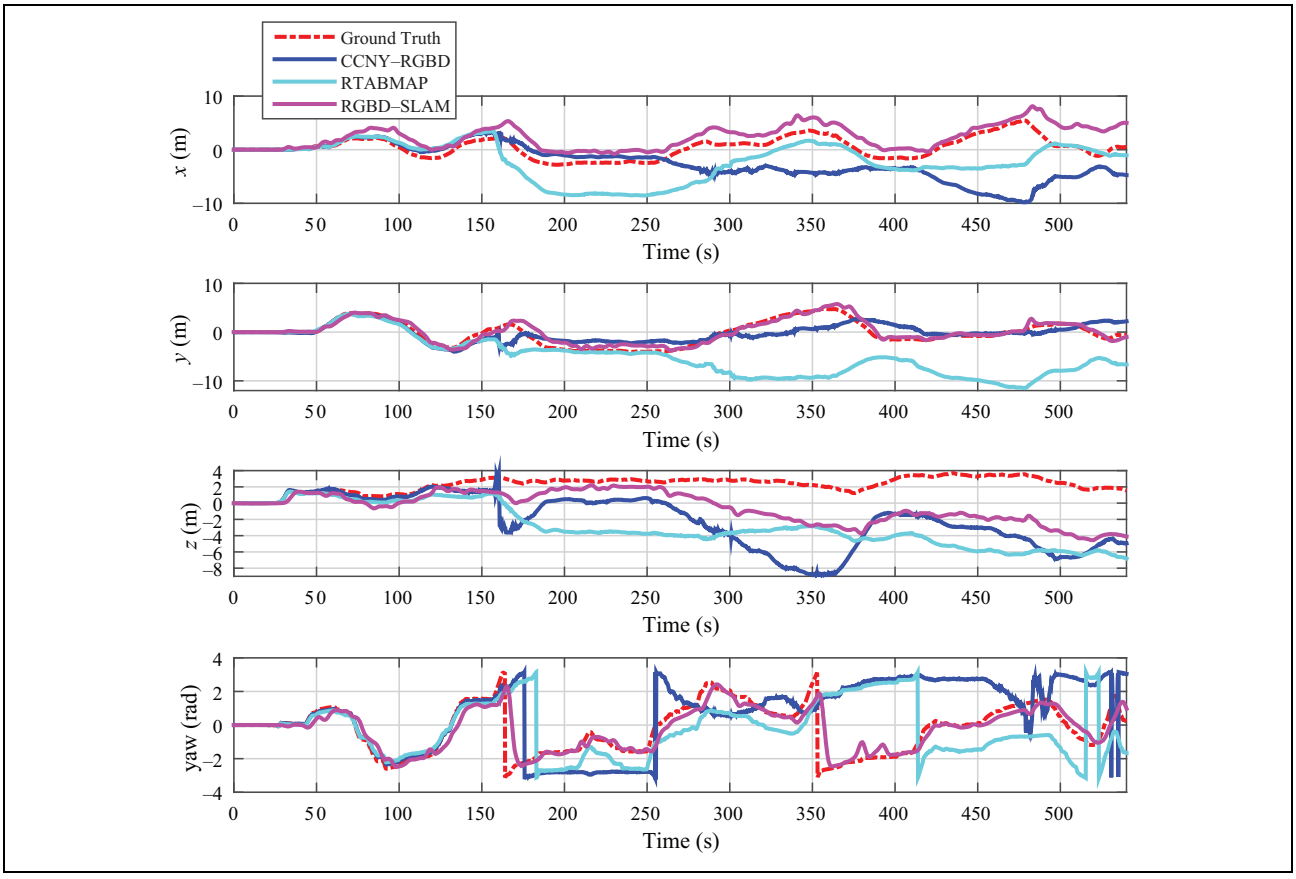

Figure 12. Estimated UAV position and orientation using other approaches based on RGB-D sensors. UAV: unmanned aerial vehicle.

Table 3. RMS error comparison.

	x (m)	y (m)	z (m)	yaw (rad)
CCNY RGBD	5.07	1.94	5.55	2.24
RTAB-Map	3.68	6.59	6.09	2.06
RGBD-SLAM	2.46	0.79	3.63	0.71
Our approach	0.36	0.39	0.17	0.18

RMS: root mean square.

Table 4. Pose estimation speed.

	Average frequency (Hz)	Frame processing time (ms)
CCNY-RGBD	8.1	124
RTAB-Map	6.2	161
RGBD-SLAM	5.8	173
Our approach	15.8	63

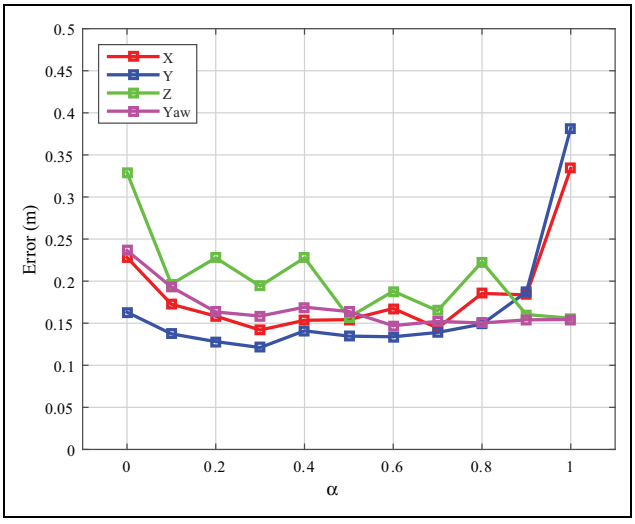

Figure 13. Localization errors for different values of α.

IMU used in our approach provides accurate and filtered estimations of such angles.

These results (Figures 10 and 11) were obtained after configuring the MCL algorithm with the parameters shown in Table 1.

The plot also shows the ground-truth position and orientation provided by the test bed's motion capture system. It can be seen how the visual odometry slowly accumulates errors over time, while the estimations from our complete approach closely follow the ground truth during all the experiments. It is also worth to mention that this

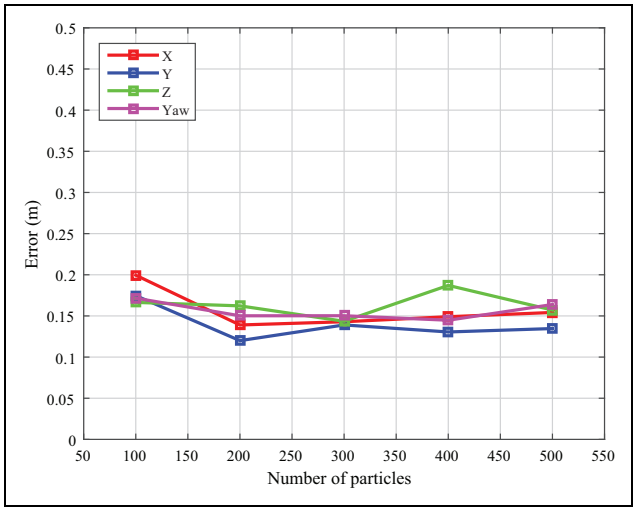

Figure 14. Localization errors for different number of particles used in MCL. MCL: Monte Carlo localization.

Table 5. Computation times for one MCL iteration when modifying the number of particles.

Number of particles	Average processing time (ms)
100	26
200	42
300	50
400	58
500	65

MCL: Monte Carlo localization.

estimation does not drift with time and the errors are approximately bounded.

Figure 11 shows the error of the estimation during the trajectory execution. The error is computed as the Euclidean distance between the estimation and the ground truth at every time step. Besides, the plot depicts the computed root mean square (RMS) errors for each axis, which are also shown in Table 2.

It can be seen how the RMS errors are approximately 0.4 m in x and y, while in z is less than 0.2 m. RMS error in yaw angle is around 10°. Errors in x and y are higher due to the larger distance traveled through these axes. Nevertheless, the global RMS error in position is 0.32 m which can be considered acceptable for robot navigation.

In order to quantitatively assess the performance of our localization approach, we have tested the data from the validation experiment against other popular approaches that make use of RGB-D data for robot localization, either visual odometry and mapping such as CCNY_RGBD (http://wiki.ros.org/ccny rgbd) or SLAM approaches such as RTAB-Map (http://wiki.ros.org/rtabmap) or RGBD-SLAM (http://wiki.ros.org/rgbdslam). The plots in Figure 12 show the performance of the same flight data processed off-line using these approaches, and Table 3 summarizes general RMS errors compared to those from our approach.

The RMS errors in x, y, and z from the compared approaches are slightly higher. Omitting z values, which could be observable through the use of an altimeter, only RGBD-SLAM seems to be a viable alternative to the proposed approach. We are able to provide a better accuracy while keeping robustness and computational efficiency as key features. Notice that we did not fine tune the parameters of the algorithms, so a better accuracy could be achieved.

Table 4 includes values for the mean frequency and corresponding mean processing times at which new pose estimations are published, using an i7 Linux laptop with 2 Cores (the UAV onboard processor is also an i7 Linux computer). Our approach, including the visual odometry, is able to deliver pose estimations approximately every 60 ms when using 500 particles, which makes it suitable for working at around half the frame rate of the RGB-D sensor. The other approaches may publish pose estimations to the ROS ecosystem at a higher rate than that of the table, but internally they just copy the same values to the output message until a new estimation is computed.

The visual odometer took an average 90% of a single thread and the proposed MCL algorithm the 55% of a single thread. This configuration leaves plenty of computation for robot planning and navigation.

The main contribution of this approach is that the localization accuracy can be improved in long-term operation using both RGB-D and radio-range sensing. Therefore, the parameter α from equation (8), which is used to combine the contributions of map matching and distances to beacons, is modified in order to assess localization errors. The rest of parameters from Table 1 remained the same for this experiment. Figure 13 shows errors in X, Y, Z, and Yaw for different values of α, from $\alpha = 0$ (MCL only relies on radio-based sensing) to $\alpha = 1$ (MCL only uses map matching). While, in general, RMS errors are low, best performances in terms of overall stability and particle cloud convergence were achieved between $\alpha = 0.5$ and $\alpha = 0.8$. As expected, Yaw error increases when the contribution from map matching decreases (lower α), since radio sensing does not provide such angle and the approach relies on the odometry integration. Errors in Z are also expected to be higher when α is closer to 0, since the spatial distribution of the installed radio beacons in height was not as much as in the XY plane, leading to poor estimations in the UAV altitude when not using the point cloud matching with the 3-D map. On the other hand, when α is close to 1, point cloud matching leads to smoother estimations but sometimes is not enough to correct odometry drift, leading to higher errors in X and Y.

Figure 14 presents the estimation accuracy using different numbers of particles, along with the required computational times included in Table 5 in order to demonstrate the capabilities of our approach. The rest

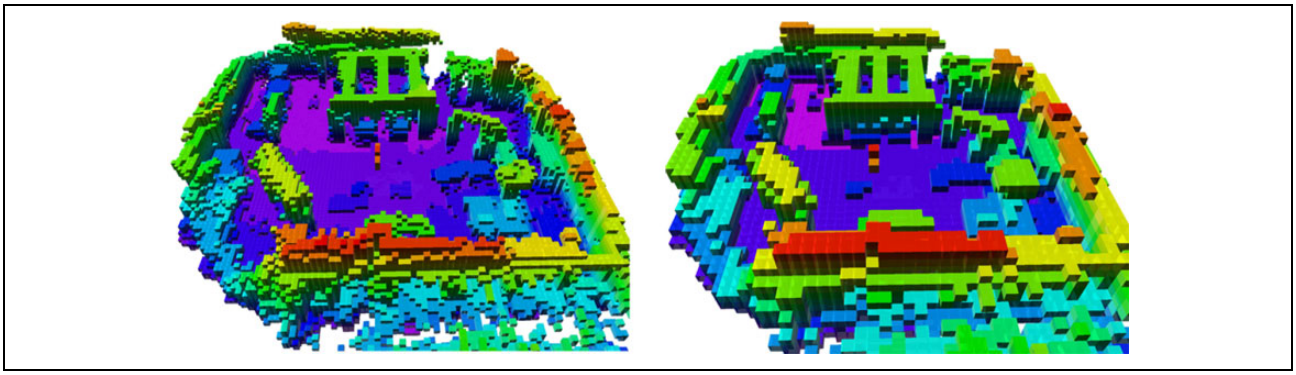

Figure 15. 3-D map of the area with different resolutions: 0.2 m (left) and 0.4 m (right).

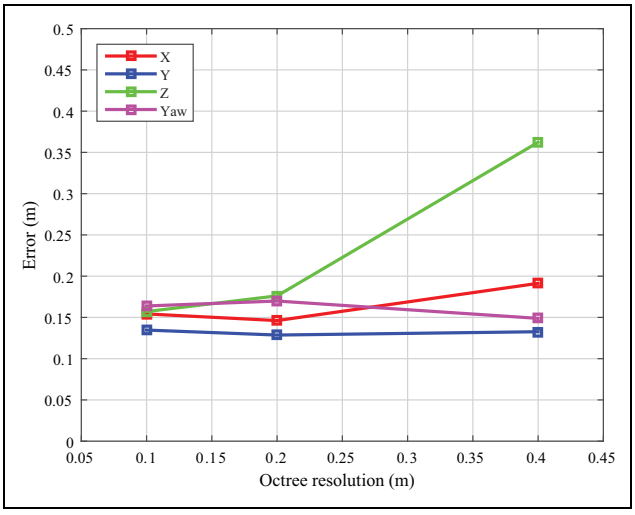

Figure 16. Localization errors for different octree resolutions of the 3-D map.

of parameters are still the default ones, as shown in Table 1. It can be seen how the impact of the number of particles on RMS errors is fairly minor, while computation times are drastically reduced.

The spatial occupancy of the working area is approximated by an octree, as previously shown in Figure 5. Another set of experiments has been performed by modifying the spatial resolution of such map from the default (0.1 m) to the following two depths of the tree, which correspond to leaf sizes of 0.2 m and 0.4 m. Figure 15 shows these maps. Running the experiment with higher sizes was discarded because the map is no longer representative enough for map matching. The number of particles was 500, and $\alpha = 0.5$. Figure 16 shows the localization RMS errors using different map resolutions. It can be noted how when using a map resolution of 0.2 m, and downsampling the sensor point clouds accordingly, it is possible to achieve similar localization errors while decreasing the computational complexity of the approach, as shown in Table 6.

Table 6. Computation times for one MCL iteration when modifying the 3-D map resolution.

Map resolution (m)	Average processing time (ms)
0.1	65
0.2	52
0.4	28

MCL: Monte Carlo localization.

Conclusions and future work

An implementation of MCL suitable for operation in 3-D environments has been presented. Visual odometry based on an RGB-D sensor is used along with a previously known map of the environment and a set of radio-based sensors for point-to-point distance measurements.

Experimental results demonstrate a strong overall system performance as well as the feasibility of the approach, both in accuracy and computational efficiency. The proposed localization approach provides a reliable and accurate 4-D (x, y, z, and ψ) estimation during the 9 min flight of the experiment. In view of these results, we think this algorithm could be used for longer periods of time, but this should be evaluated in the future with longer experiments.

Future work will consider the inclusion of other sensor modalities into the localization system. Information as visual place recognition or altimeter can be used to draw new particles into the filter. Additionally, the authors will also evaluate how to compute accurate 3-D maps of the environment including range sensors to be used as input to the proposed approach. Moreover, improvements to the conventional MCL used in this approach will be evaluated, such as modifying the resampling method while keeping in mind the computational efficiency of the whole algorithm.

Finally, the authors are currently preparing to release the source code of the algorithms and add the corresponding information on the ROS wiki to be publicly available.

Acknowledgements

This research is a result of a partnership between the University of Seville and the Center for Advanced Aerospace Technologies (CATEC). Thanks also go to Angel Luis Petrus.

Declaration of conflicting interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This work has been partially supported by EuRoC project funded by the EU FP7 under grant agreement CPIP 608849 and OCELLI-MAV project funded by the Spanish Government, grant TEC2014-61708-EXP.

References

1. Cox IJ. Blanche—an experiment in guidance and navigation of an autonomous robot vehicle. *IEEE Trans Robot Autom* 1991; 7(2): 193–204.
2. Marder-Eppstein E, Berger E, Foote T, et al. The office marathon: robust navigation in an indoor office environment. In: *2010 IEEE international conference on robotics and automation (ICRA)*, Anchorage, AK, pp. 300–307. IEEE.
3. Montemerlo M, Becker J, Bhat S, et al. Junior: the Stanford entry in the urban challenge. *J Field Robot* 2008; 25(9): 569–597.
4. Thrun S. *Sebastian Thrun: Google's driverless car*. TED Talks, 2011.
5. Siebert S and Teizer J. Mobile 3D mapping for surveying earthwork projects using an unmanned aerial vehicle (UAV) system. *Autom Constr* 2014; 41: 1–14.
6. Bolognesi M, Furini A, Russo V, et al. Testing the low-cost RPAS potential in 3D cultural heritage reconstruction. *Int Arch Photogramm Remot Sens Spatial Inf Sci* 2015; 40(5): 229.
7. Mesas-Carrascosa FJ, Notario-García MD, de Larriva JEM, et al. Validation of measurements of land plot area using UAV imagery. *Int J Appl Earth Obs Geoinf* 2014; 33: 270–279.
8. Raja AK and Pang Z. High accuracy indoor localization for robot-based fine-grain inspection of smart buildings. In: *2016 IEEE international conference on industrial technology (ICIT)*, Taipei, pp. 2010–2015. IEEE.
9. Cui JQ, Phang SK, Ang KZ, et al. Search and rescue using multiple drones in post-disaster situation. *Unmanned Syst* 2016; 4(01): 83–96.
10. Lee KS, Ovinis M, Nagarajan T, et al. Autonomous patrol and surveillance system using unmanned aerial vehicles. In: *2015 IEEE 15th international conference on environment and electrical engineering (EEEIC)*, Rome, pp. 1291–1297. DOI: 10.1109/EEEIC.2015.7165356.
11. Davison A, Reid I, Molton N, et al. MonoSLAM: real-time single camera SLAM. *IEEE Trans Pattern Anal Mach Intell* 2007; 29(6): 1052–1067. DOI: 10.1109/TPAMI.2007.1049.
12. Weiss S, Scaramuzza D, and Siegwart R. Monocular-SLAM-based navigation for autonomous micro helicopters in GPS-denied environments. *J Field Robot* 2011; 28(6): 854–874. DOI: 10.1002/rob.20412.
13. Pinies P, Lupton T, Sukkarieh S, et al. Inertial aiding of inverse depth SLAM using a monocular camera. In: *2007 IEEE international conference on robotics and automation*, Roma, pp. 2797–2802. DOI: 10.1109/ROBOT.2007.363895.
14. Achtelik M, Lynen S, Weiss S, et al. Visual-inertial SLAM for a small helicopter in large outdoor environments. In: *2012 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, Vilamoura, pp. 2651–2652. DOI: 10.1109/IROS.2012. 6386270.
15. Pagliari D, Cazzaniga NE, and Pinto L. Use of assisted photogrammetry for indoor and outdoor navigation purposes. In: *ISPRS—international archives of the photogrammetry, remote sensing and spatial information sciences*, 2015, pp. 113–118. DOI: 10.5194/isprsarchives-XL-4-W5.
16. Geiger A, Ziegler J, and Stiller C. StereoScan: dense 3D reconstruction in real-time. In: *2011 IEEE intelligent vehicles symposium (IV)*, Baden-Baden, pp. 963–968. DOI: 10.1109/IVS. 2011.5940405.
17. Schmid K, Tomic T, Ruess F, et al. Stereo vision based indoor/outdoor navigation for flying robots. In: *2013 IEEE/RSJ international conference on intelligent robots and systems*, Tokyo, pp. 3955–3962. DOI: 10.1109/IROS.2013.6696922.
18. Kitt B, Geiger A, and Lategahn H. Visual odometry based on stereo image sequences with RANSAC-based outlier rejection scheme. In: *Intelligent vehicles symposium (IV)*, San Diego, CA, 2010, pp. 486–492.
19. Endres F, Hess J, Engelhard N, et al. An evaluation of the RGB-D SLAM system. In: *2012 IEEE international conference on robotics and automation (ICRA)*, Saint Paul, MN, 2012, pp. 1691–1696. DOI: 10.1109/ICRA.2012.6225199.
20. Kerl C, Sturm J, and Cremers D. Robust odometry estimation for RGB-D Cameras. In: *2013 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3748–3754. IEEE.
21. Dryanovski I, Valenti RG, and Xiao J. Fast visual odometry and mapping from RGB-D data. In: *2013 IEEE international conference on robotics and automation*, Karlsruhe, 2013, pp. 2305–2310. DOI: 10.1109/ICRA.2013.6630889.
22. Labbe M and Michaud F. Online global loop closure detection for large-scale multi-session graph-based SLAM. In: *Proceedings of the IEEE/RSJ international conference on intelligent robots and systems*, Chicago, IL, 2014, pp. 2661–2666.
23. Lowry S, Sünderhauf N, Newman P, et al. Visual place recognition: a survey. *IEEE Trans Robot* 2016; 32(1): 1–19.
24. Sünderhauf N and Protzel P. BRIEF-Gist—closing the loop by simple means. In: *2011 IEEE/RSJ international conference on intelligent robots and systems*, San Francisco, CA, 2011, pp. 1234–1241. DOI: 10.1109/IROS.2011.6094921.
25. Endres F, Hess J, Sturm J, et al. 3-D mapping with an RGB-D camera. *IEEE Trans Robot* 2014; 30(1): 177–187.
26. Chen Z and Birchfield ST. Qualitative vision-based mobile robot navigation. In: *ICRA 2006. Proceedings 2006 IEEE*

international conference on robotics and automation, Orlando, FL, 2006, pp. 2686–2692. IEEE.
27. Royer E, Lhuillier M, Dhome M, et al. Monocular vision for mobile robot localization and autonomous navigation. *Int J Comput Vis* 2007; 74(3): 237–260.
28. Thrun S, Fox D, Burgard W, et al. Robust Monte Carlo localization for mobile robots. *Artif Intell* 2001; 128(1): 99–141. DOI: 10.1016/S0004-3702(01)00069-8.
29. Hornung A, Wurm KM, and Bennewitz M. Humanoid robot localization in complex indoor environments. In: *Proceedings of the IEEE/RSJ international conference on intelligent robots and systems (IROS)*, Taipei, Taiwan, October 2010, pp. 1690–1695.
30. Kanai S, Hatakeyama R, and Date H. Improvement of 3d Monte Carlo localization using a depth camera and terrestrial laser scanner. *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 40(4): 61.
31. Fallon MF, Johannsson H, and Leonard JJ. Efficient scene simulation for robust Monte Carlo localization using an RGBD camera. In: *2012 IEEE international conference on robotics and automation (ICRA)*, Saint Paul, MN, 2012, pp. 1663–1670. IEEE.
32. Gholami M, Cai N, and Brennan R. An artificial neural network approach to the problem of wireless sensors network localization. *Robot Comput Integr Manuf* 2013; 29(1): 96–109.
33. Fabresse F, Caballero F, Maza I, et al. Robust range-only SLAM for unmanned aerial systems. *Robot J Intell Robot Syst* 2016; 84(1–4): 297–310.
34. Caballero F, Merino L, Maza I, et al. A particle filtering method for wireless sensor network localization with an aerial robot beacon. In: *ICRA 2008. IEEE international conference on robotics and automation*, Pasadena, CA, 2008, pp. 596–601. DOI: 10.1109/ ROBOT.2008.4543271.
35. González J, Blanco JL, Galindo C, et al. Mobile robot localization based on ultra-wide-band ranging: a particle filter approach. *Robot Auton Syst* 2009; 57(5): 496–507.
36. Tiemann J, Schweikowski F, and Wietfeld C. Design of an UWB indoor-positioning system for UAV navigation in GNSS-denied environments. In: *2015 international conference on indoor positioning and indoor navigation (IPIN)*, Banff, AB, 2015, pp. 1–7. IEEE.
37. Guo K, Qiu Z, Miao C, et al. Ultra-wideband-based localization for quadcopter navigation. *Unmanned Syst* 2016; 4(01): 23–34.
38. Perez-Grau FJ, Fabresse FR, Caballero F, et al. Long-term aerial robot localization based on visual odometry and radio-based ranging. In: *2016 international conference on unmanned aircraft systems (ICUAS)*, Arlington, VA, 2016, pp. 608–614. DOI: 10.1109/ICUAS. 2016.7502653.
39. Bouabdallah S and Siegwart R. *Design and control of a miniature quadrotor, intelligent systems, control and automation: science and engineering*, vol. 33. Dordrecht: Springer, 2007.
40. Hornung A, Wurm KM, Bennewitz M, et al. OctoMap: an efficient probabilistic 3D mapping framework based on octrees. *Auton Robot* 2013; 34(3): 189–206.
41. Thrun S. Probabilistic robotics. *Communications of the ACM*, 45(3): 52–57.