DOI: 10.1515/aon-2018-0010

PIOTR KANIEWSKI, PAWEŁ SŁOWAK
Military University of Technology, Warszawa, Poland

SIMULATION AND ANALYSIS OF PARTICLE FILTER BASED SLAM SYSTEM

ABSTRACT

The paper describes a problem and an algorithm for simultaneous localization and mapping (SLAM) for an unmanned aerial vehicle (UAV). The algorithm developed by the authors estimates the flight trajectory and builds a map of the terrain below the UAV. As a tool for estimating the UAV position and other parameters of flight, a particle filter was applied. The proposed algorithm was tested and analyzed by simulations and the paper presents a simulator developed by the authors and used for SLAM testing purposes. Chosen simulation results, including maps and UAV trajectories constructed by the SLAM algorithm are included in the paper.

Keywords:
SLAM, particle filter, simulation, INS, monocular camera

1. INTRODUCTION

A wide spectrum of complex tasks for modern robotics systems demands that various robotic platforms should be increasingly self-sufficient. The requirement of autonomy is often imposed on unmanned aerial vehicles (UAVs) flying beyond the line of sight of their operators. Autonomous UAVs often must be able to operate in unknown or even dynamically changing environment and apart from estimating their position be capable of correct interpretation of the surroundings. This is a problem identified in the technical literature as simultaneous localization and mapping (SLAM) [Thrun et al., 2005].

SLAM consists in concurrent estimation of locations of characteristic features of the environment (landmarks) and the state of the robot, including its position, veloci-

ty and orientation, using autonomous on-board sensors. Solving the SLAM problem can be divided into several tasks, each requiring a distinct approach, such as landmark extraction, data association, state estimation and map update.

SLAM capable platforms offer two main advantages. Firstly, the robot remains fully operational without the use of external signals, e.g. from Global Navigation Satellite Systems (GNSS). This allows it to explore areas with limited availability of GNSS, e.g. indoors or contemporary battlefields where use of jamming is highly probable. Moreover, a SLAM robot performs a constant environment analysis, identifying landmarks locations and combining them into a globally consistent map. The map is later used in robot's navigation process. Thus, SLAM platforms can operate in previously unvisited surroundings and even distribute the acquired map among the cooperating UAVs [Howard, 2006].

However, the SLAM intrinsic feature of interdependency between the need of precise robot's location to acquire map and the need of a precise map to properly estimate its location is a complex problem to address. During last 30 years of SLAM development, there were several characteristic approaches to solve concurrent localization and mapping problem. To properly list various technics used, it is necessary to divide SLAM architecture into a front-end and a back-end component. The front-end component consists in landmarks extraction whereas the back-end one is responsible for map and localization estimation.

Initially, extraction of features was performed with the use of data from sonar sensors or laser rangefinders [Leonard at al., 1992, Bailey at al., 2006]. Since the computational complexity of image processing became feasible for modern computers, in recent years, various types of cameras, such as monocular camera [Santana et al., 2011] or RGB-D camera [Endres et al., 2014] became popular tools for landmark extraction. Many authors adopted different feature extraction algorithms suitable for processing images in SLAM. The most commonly used algorithms are SIFT and SURF, which are based on gradient histogram feature descriptors or their less computationally expensive alternatives such as BRIEF, ORB, BRISK and FREAK [Hartman et al., 2013], based on binary feature descriptors.

The back-end component was implemented in three most common ways over the years. The earliest approaches were constructed upon Extended Kalman Filter (EKF) [Leonard at al., 1990]. Due to its poor approximation of strongly nonlinear functions, other solutions were proposed like unscented Kalman filter [Wang et al., 2013]) or particle filter [Thrun et al., 2002]. Lately, graph-based methods have been introduced, which rely on least-squares error minimization [Grisetti et al., 2010].

Concurrent localization and mapping, due to its complexity, is still an open research field for new approaches, including more robust and more efficient algo-

rithms [Cadena et al., 2016]. From the authors' perspective, it is also a very important and interesting problem due to its potential significance for military applications, as it is predicted that future military missions will increasingly rely on autonomous platforms, including UAVs or large formations (swarms) of cooperating UAVs [Vincent et al., 2004].

The further part of the paper is organized as follows. The section 2 describes the SLAM algorithm, which was implemented by the authors with the use of MATLAB®. A SLAM system simulator used for testing this algorithm is presented in section 3. The section 4 contains chosen simulation results, which are concluded in section 5.

2. SLAM ALGORITHM

The front-end part of the SLAM algorithm elaborated by the authors uses SURF algorithm to extract features from images. It is assumed that the images are provided by a monocular camera installed on board a UAV and directed downwards. The back-end component of the algorithm is based upon a particle filter. The particle filter is a non-parametric estimation algorithm, which uses a set of samples (particles) to represent the full probability distribution function (PDF) of the UAV state vector and the map elements [Konatowski et al., 2016]. The flowchart of SLAM algorithm is shown in Fig. 1.

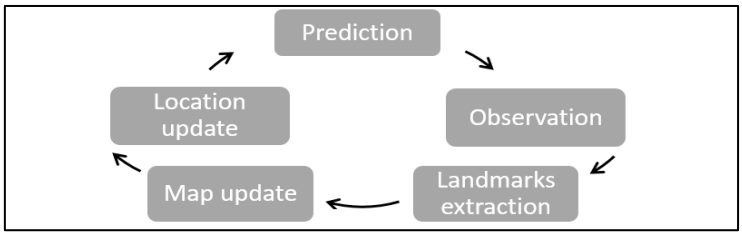

Fig. 1. Flowchart of the SLAM algorithm [Thrun et al., 2005].

The SLAM algorithm is implemented as a loop composed of a series of steps. Each iteration of the loop consists of UAV state prediction, image acquisition from the camera (observation), landmarks extraction using SURF algorithm, map update and robot's localization update.

The first step of the presented algorithm consists in prediction of the new localization of the UAV. The state vector to be estimated in this process consists of nine

quantities: three for position, three for velocity and three for orientation. The above robot's state is represented in our algorithm by a large set of particles, which can be considered as many possible realizations of the state vector, spread-out in accordance with the PDF. We assume that the UAV is equipped into an Inertial Measurement Unit (IMU) providing measurements of linear accelerations and angular velocities. These measurements are composed of true quantities and errors, which we model as Gaussian white noises. In the prediction process, the particle filter algorithm generates as many realizations of these errors as many particles exist. The generated errors are added to the linear accelerations and angular velocities from IMU and the obtained quantities are control signals treated as inputs to a strap-down inertial navigation system (SINS) algorithm [Łabowski et al. 2016]. Thus, each particle receives different controls and we obtain many distinct particles which describe PDF of the new predicted robot's state. This is an implementation of the particle filter prediction step, which can be expressed as follows:

$$x_{k+1}^{(i)} \sim p\left(x_{k+1} \middle| x_k^{(i)}, u_k\right) \tag{1}$$

where $x_k^{(i)}$ is the state vector of the i^{th} particle during $k^{\text{-th}}$ step and u_k are controls.

Afterwards, an observation takes place and a fragment of terrain, registered by the UAV monocular camera, is analyzed. This is illustrated in Fig. 2.

Fig. 2. Illustration of the observation process using a monocular camera installed on UAV.

The aim of observation is to find characteristic features, that can be used for relative UAV localization as well as map construction. Those features are commonly

called landmarks. Speeded Up Robust Features (SURF) algorithm was chosen for searching landmarks in the image. It was selected as it is relatively fast, scale invariant, robust against image transformation and reliable in terms of repeatability [Baya et al., 2008]. SURF procedure was developed to find correspondences between images. The algorithms for feature matching are commonly described in terms of two basic components. The first one is a detector of interest points and the second one is a feature descriptor.

The SURF detector determines image pixels around which there are the most significant intensity changes. The selection is based on finding the local maxima of Hessian matrix determinant according to the expression [Baya et al., 2008]:

$$det(H_{approx}) = D_{xx}D_{yy} - (0.9D_{xy})^2$$

where D_{ii} are box filters used for an approximation of second-order scale-normalized Gaussian kernels. The box filters used in the detector are shown in Fig. 3.

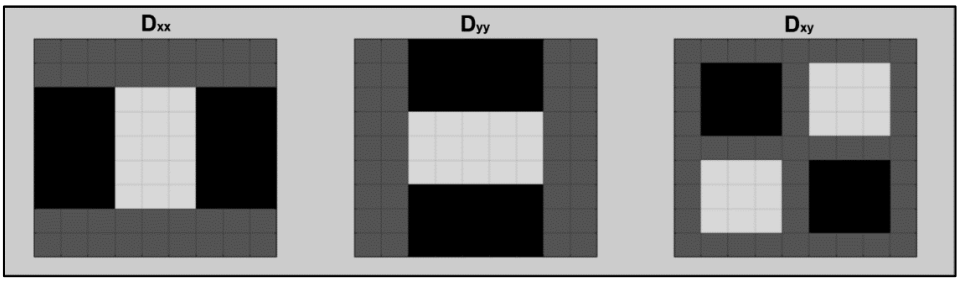

Fig. 1. Box filters in the x, y and xy directions.

The invariance of feature detection in terms of scale is achieved by examining image at different scales. It is commonly performed by repeated image convolution with box filters of various sizes. As a result, the SURF algorithm indirectly performs an analysis of scaled image while performing calculations of the detector simultaneously.

The SURF descriptor is defined by an intensity distribution of pixels in the vicinity of a point that was selected by the detector. It has a form of a 64-element vector which is further used by the SLAM algorithm to match corresponding features in distinct images.

On each iteration, the SLAM algorithm is required to point out a predefined number of landmarks characterized by the strongest metric as defined in Eq. (2). The algorithm makes also sure that the selected landmarks do not lie too close to each

other as fragments of the image around landmarks are later used to build a map. If the default SURF parameters are too strict to extract enough landmarks, the algorithm lowers the detection threshold, so that no large blank spaces are left on the map.

After landmarks are extracted, there is a need of accurate data association. The SLAM algorithm uses described above SURF descriptor to match landmarks. Additionally, the algorithm confronts the result of descriptors matching against the global coordinates of a given landmarks pair. It minimizes a possibility of finding a false positive correspondence for similar looking but distant features. If a landmark is not matched to any previously observed, the algorithm initializes it as a new landmark.

After the association of landmarks, SLAM algorithm uses particle filter to update the map and the UAV state vector. To simplify filter complexity, Rao-Blackwellization has been implemented [Thrun et al., 2002]. Rather than computing the joint a posteriori PDF:

$$p(x_{k+1}, m_{k+1}|x_k, u_k, z_{k+1}) \tag{3}$$

about the state vector *x* and the map *m* (a set of discovered landmarks), this approach marginalizes the map out of the state space, which leads to the following factorization:

$$p(x_{k+1}, m_{k+1}|x_k, u_k, z_{k+1}) = p(x_{k+1}|x_k, u_k)p(m_{k+1}|x_{k+1}, z_{k+1}) \tag{4}$$

This way the particle filter is only used for robot's state calculation while a number of Kalman Filters (KFs) manage the calculation of landmarks location for each particle [Thrun et al., 2002]. Only one map is assigned to a given particle. Marginalizing a subset of the state space limits the number of particles needed. Rao-Blackwellization allows to omit a situation where tracking every particle's position, while considering every possible landmark location, increases the number of particles exponentially over time.

For all visible landmarks which were identified as previously observed the algorithm commences two fundamental procedures. Firstly, for each particle, the positions of all these landmarks are estimated by individual KFs, under assumption that the particle state contains the true location and orientation of UAV. This way the previous landmarks positions, stored in the map, are updated.

The second step, directly related to the robot's state update, is calculating particles weights according to the current mismatch between observed and predicted

landmarks locations. The weights are inversely proportional to the difference between these locations. The particle weights are given by the following expression:

$$w_{k+1}^{(i)} \sim p\left(y_{k+1} \middle| x_{k+1}^{(i)}\right) \tag{5}$$

where $w_{k+1}^{(i)}$ is a new weight of an i^{th} particle and y_{k+1} contains residuals, i.e. differences between the observed landmarks locations and their predicted values. After calculation of weights, their values are normalized so that their sum is one. This way the weighted particles can be considered a discrete representation of a posteriori PDF.

To complete the particle filter routine, after weighting each particle and normalizing their weights, the algorithm should resample the particles. The resampling is based on substituting the particles of lowest weight by new instances of particles of highest weight. As the operation is computationally demanding in terms of memory usage, a rule limiting the occurrence of resampling operations should be imposed. In our algorithm, the so called adaptive resampling is used [Grisetti et al., 2005]. The algorithm resamples only during iterations in which the effective number of particles, given by the equation:

$$N_{eff} = \frac{1}{\sum_{i=1}^{N} (w^i)^2} \tag{6}$$

falls below a predefined fraction of all the samples.

3. SIMULATOR OVERVIEW

In order to analyze the properties of the described SLAM algorithm, the authors developed a simulator using MATLAB computing environment. This software offers a wide variety of data analysis tools and significantly facilitates the system building process. Its usefulness for SLAM testing was previously demonstrated by other researchers [Joan Sola et al., 2009].

The prepared MATLAB application is responsible for simulation of robot (UAV) motion, generation of environment and images visible by a monocular camera, performing SLAM routine and graphical presentation of its results. For the sake of reduction of computational complexity, it is based on several simplifying assumptions. The UAV is modeled as a point in 3D space and its movement has only three de-

grees of freedom. Its motion is constrained only to a horizontal plane located on a constant altitude of 100 meters over the terrain. The simulator calculates consecutive robot positions and orientations on the basis of horizontal linear accelerations and angular velocities around the vertical axis, as predefined by the user. To model the surface below the robot we use concatenated Google Maps images [Google, 2018] and assume that the terrain is flat. The camera on board UAV points downwards, is monochrome and has QVGA resolution, i.e. 320x240 pixels.

Implementation of the particle filter is based on nested structures. The filter contains an array of particles and an array of their weights. Each particle encapsulates UAV state vector and an array of landmark objects. To limit data complexity and memory usage, landmark objects consist only of landmark positions and orientations of the robot during the observation. The landmarks SURF descriptors and image fragments around the landmarks are stored separately and shared by all the particles.

To facilitate the analysis of the SLAM algorithm the simulator generates several graphical objects. The first object illustrates the process of observation of the environment using a monocular camera and its example is shown in Fig. 2.

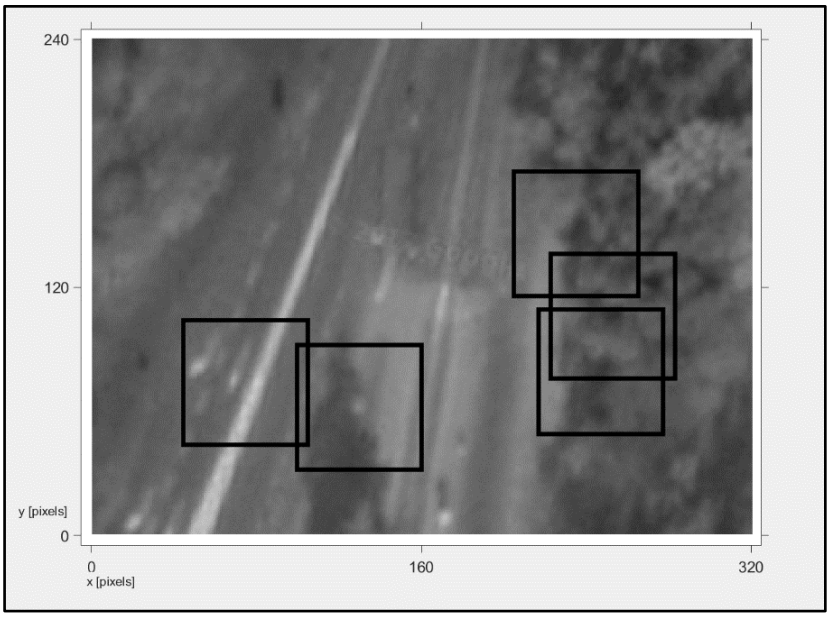

Fig. 2. Camera's view with SURF landmarks.

Another figure generated by the simulator, shown in Fig. 4, presents the image as seen by the sensor. Furthermore, currently observed landmarks and the surface fragments which are later used for map building (black squares) are also marked.

The third graphical object is a figure used to visualize particles distribution and weights. A size of each particle is proportional to its weight. An example of this graphical object, obtained during simulation, is shown in Fig. 5.

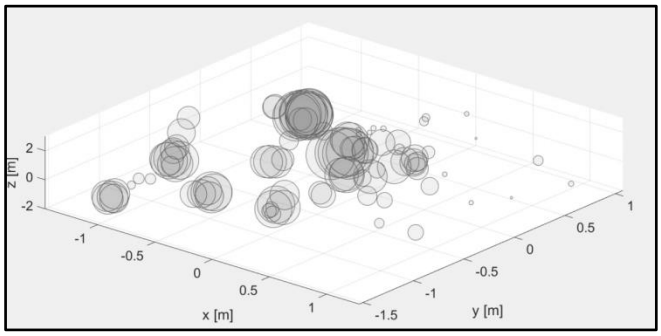

Fig. 3. Visualization of particles distribution.

The last figure, shown in Fig. 6, presents the map constructed by the SLAM algorithm. During every iteration, the current particle of the highest weight is chosen. Terrain fragments observed around the landmarks are arranged according to the coordinates provided by this particle. At the end of each loop of the SLAM algorithm, all figures are updated.

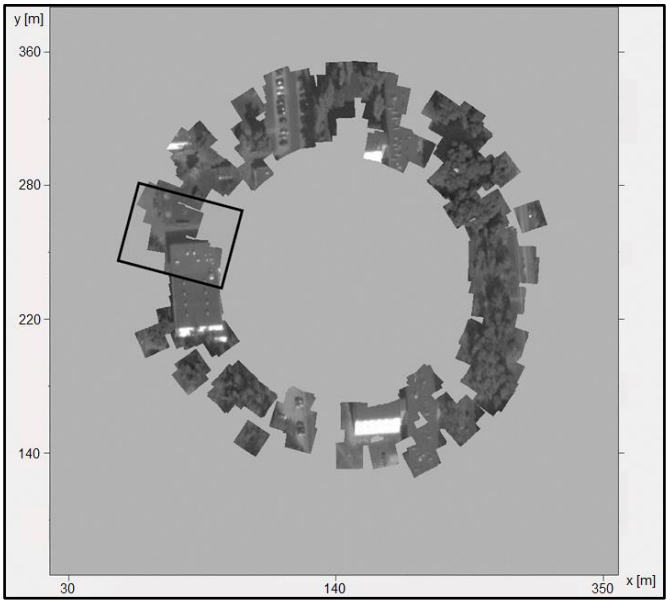

Fig. 4. Map building process.

4. RESULTS

The simulator performance was examined by assessing whether it is able to imitate SLAM algorithm as well as visualize its key features. The analyses of particles behavior, landmark matching, map building as well as overall performance were conducted.

Fig. 7-10 illustrate how the PDF of the UAV state vector, represented by a set of 100 particles, changes over time. Fig. 7 shows the particles set during the initial phase of the simulation. The particles have similar sizes as the PDF is distributed almost uniformly among the samples. The effective number of particles is slightly smaller than the true number of particles in the set.

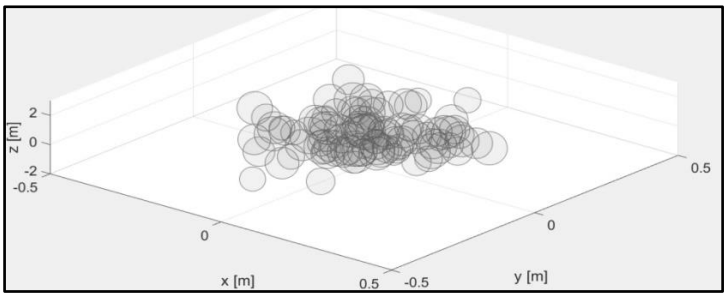

Fig. 5. Initial distribution of particles in the process of UAV state vector estimation.

As the SLAM algorithm continues to estimate the changing UAV state vector, the particles weights become increasingly varied. This implies that some of the samples are approximating the robot's state better and other samples are less probable estimates in a filtering process. The effective number of particles decreases until it drops below the resampling threshold of 50 particles. Fig. 8 illustrates the last step of the algorithm before the resampling. Sizes of the particles differ noticeably and their spread is larger than in Fig. 7.

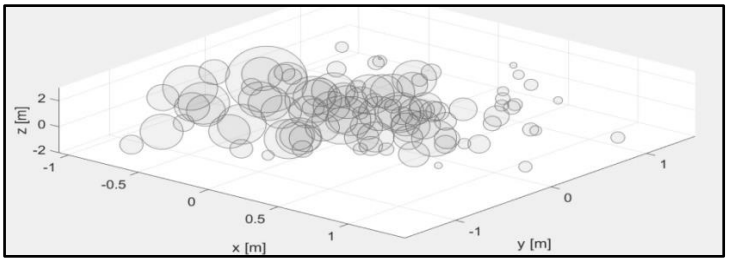

Fig. 6. Distribution of particles before the resampling.

The distribution of particles after the resampling is presented in Fig. 9. The particles which had the largest weights prevailed and replaced the samples representing less probable estimates. Due to the resampling process, the effective number of particles increases.

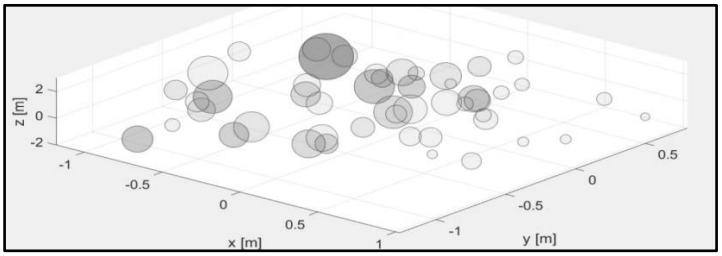

Fig. 7. Distribution of particles after the resampling.

Fig. 10 illustrates how, after the resampling, the particles start again to diverge in various directions. Consequently, the effective number of particles decreases and the whole procedure is repeated. The analysis shows that particles behave in accordance with theoretical assumptions.

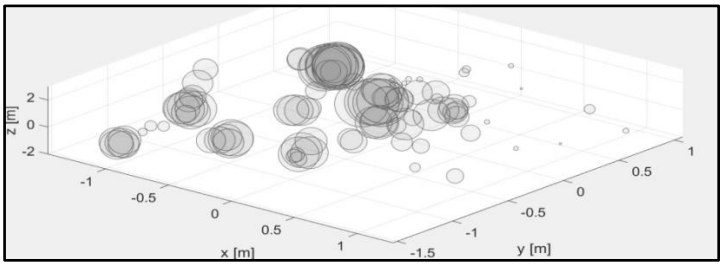

Fig. 8. Distribution of particles in further steps after the resampling.

Another expected feature of our system is an ability of proper landmarks identification. Fig. 11 shows how the number of unique landmarks increases during simulation. After the UAV reaches a previously known area, the ratio at which the number of unique landmarks is growing decreases more than three times. This indicates that while a small number of new features is still detected, most of the landmarks are correctly identified as already known. Using the feature detector and descriptor provided by the SURF algorithm, proves to be sufficient for unambiguous landmarks matching.

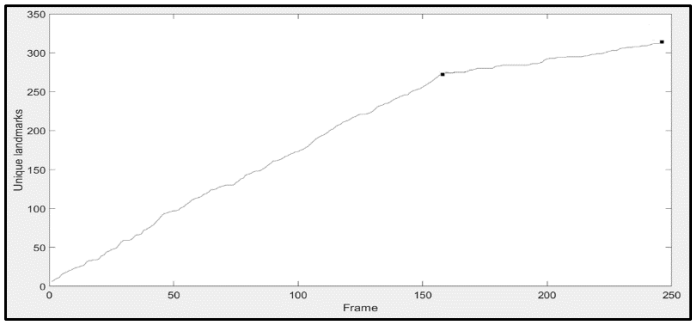

Fig. 9. Number of registered unique landmarks in time.

The assessment of map building process was performed by examining the alignment of camera image fragments extracted together with landmarks in consecutive observations. Fig. 12 shows such image fragments arranged in accordance with the UAV position. No noticeable mismatches of the fragments can be seen. This confirms that they were adequately aligned in the figure representing the map and proves that coordinates transformations throughout different reference frames as well as landmarks associations were performed properly by the SLAM algorithm.

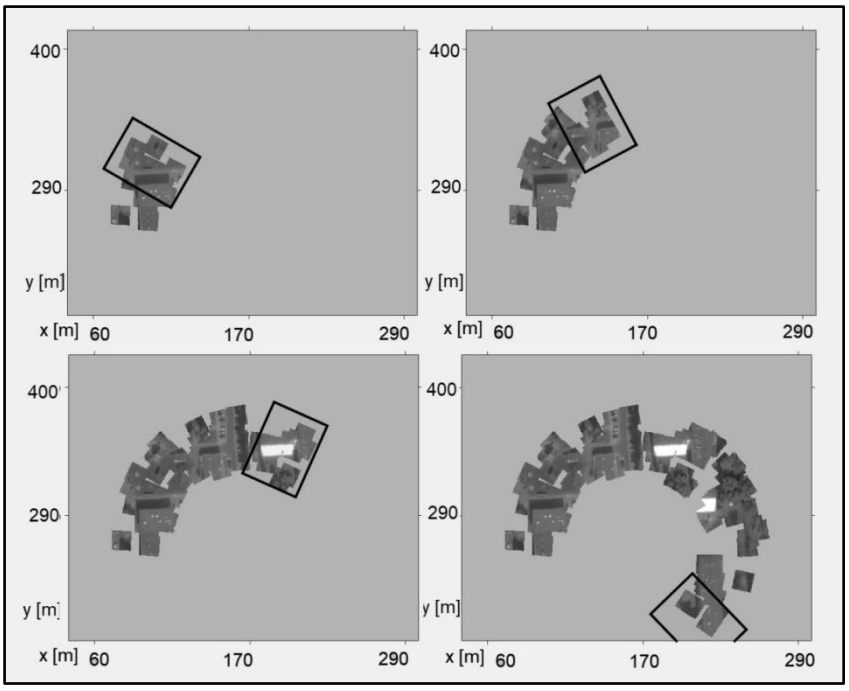

Fig. 10. Consecutive steps of map building with camera field of view.

Overall performance of the simulator was evaluated by analysing the trajectories of particles relatively to the UAV path. Fig. 13 presents the map constructed by the robot as it moves along its trajectory (black line) together with the particles trajectories (white lines). From the beginning of simulation, the particles drift increases gradually. This results from an intrinsic property of a strapdown inertial navigation system which integrates measurements of accelerometers and gyros, leading also to an integration of errors. However, if a particle current location differs significantly from the true UAV position, observation errors appear. Consequently, the particle weight decreases, making it much less probable for the particle to survive the resampling process. This way the algorithm ensures that its particles remain in the vicinity of the robot's true trajectory. As can be seen in Fig. 13, the SLAM algorithm correctly estimates the UAV path.

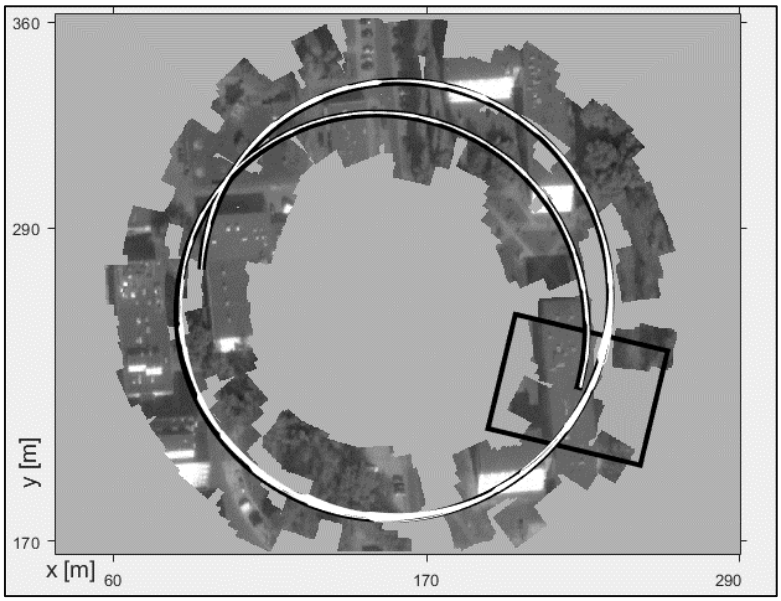

Fig. 11. UAV's true trajectory (black) and particles' estimation of it (white).

Further, an analysis of the algorithm's behavior when the UAV returns to a previously visited area was performed. Then, the SLAM algorithm should recognize already registered landmarks causing the accuracy of estimation to improve. This phenomenon is addressed as closing the loop in the SLAM terminology. Fig. 14 shows an example of the loop closing. Before visiting an already known area (Fig. 14a) the particles are shifted from the true UAV trajectory and distributed relatively wide. After the UAV reaches a previously visited region (Fig. 14b) the shift and

spread of the particles significantly decrease, indicating an improvement of the particle filter accuracy. This results from the fact, that in the previously visited area, samples are weighted not only relatively to few landmarks from several recent frames, but also relatively to the landmarks extracted during previous visits.

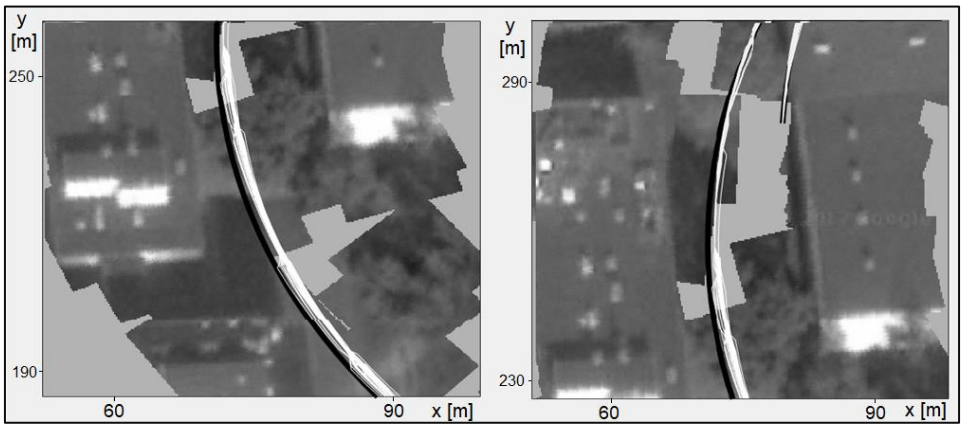

Fig. 12. Visiting known area and loop closing.

5. CONCLUSIONS

In the paper, a SLAM algorithm based on a Rao-Blackwellized particle filter and processing images from a monocular camera, dedicated for use in UAVs, was presented. The algorithm was implemented in MATLAB computing environment and thoroughly tested with a simulator developed by the authors. The simulator made it possible to examine key features of the proposed SLAM algorithm, including the particle filter performance, the effectiveness of SURF features extraction and matching as well as its ability to build a map of the terrain during UAV flight. The behaviour of our SLAM algorithm proved to be correct at all its stages of operation which is a claim supported by the simulation results included in the paper.

While the developed simulator is a flexible and useful tool, there is still a space for its upgrade. Some of the features which are planned to be included by the authors are:

- improving motion and observation models to allow for a six degrees of freedom analyses;
- expanding the simulator to a multi-robot SLAM capability;
- adding a possibility of importing and processing data acquired by a real UAV.

The developed simulator gives a possibility of gaining experience needed to design a fully operational outdoor SLAM system for large-scale experiments. However, the proposed approach can be fully validated only using data acquired by real sensors installed onboard UAV. The authors currently develop a SLAM capable UAV demonstrator and plan to verify the presented simulation results using real registered IMU data and camera images.

REFERENCES

[1] Bailey T., Durrant-Whyte H.: Simultaneous localization and mapping (SLAM): part II. IEEE Robotics & Automation Magazine, Vol. 13, Issue 3, 2006, pp. 108-117.

[2] Baya H., Essa A., Tuytelaarsb T., Van Goolab L. (2008), Speeded-Up Robust Features (SURF). Computer Vision and Image Understanding, Vol. 110, Issue 3, 2008, pp. 346-359.

[3] Cadena C., Carlone L., Carrillo H., Latif Y., Scaramuzza D., Neira J., Reid I., Leonard J.: Past, Present, and Future of Simultaneous Localization and Mapping: Towards the Robust-Perception Age. IEEE Transactions on Robotics, Vol. 32, issue 6, 2016.

[4] Durrant-Whyte H., Leonard J., Cox J.I.: Dynamic map building for autonomous mobile robot. Intelligent Robots and Systems '90, 1990.

[5] Endres F., Hess J., Sturm J., Cremers D., Burgard W.: 3D Mapping with an RGB-D Camera. IEEE Transactions on robotics Vol. 30, issue 1, 2010, pp. 177-187.

[6] Grisetti G., Kummerle R., Stachniss C., Burgard W.: A Tutorial on Graph-Based SLAM. IEEE Intelligent Transportation Systems Magazine, Vol. 2, issue 4, 2010, pp. 31-43.

[7] Grisetti G., Stachniss C., Burgard W.: Improving Grid-based SLAM with Rao-Blackwellized Particle Filters by Adaptive Proposals and Selective Resampling. Proceedings of the 2005 IEEE International Conference on Robotics and Automation Barcelona, Spain, 2005, pp. 2432-2437.

[8] Hartmann J., Klussendorff J., Maehle E.: A comparison of feature descriptors for visual SLAM, European Conference on Mobile Robots 2013.

[9] Howard A.: Multi-robot Simultaneous Localization and Mapping using Particle Filters. Proceedings of the 2005 IEEE International Conference on Robotics and Automation.

[10] Leonard J.J., Durrant-Whyte H.F.: Directed Sonar Sensing for Mobile Robot Navigation. Kluwer Academic Publishers, 1992.

[11] Łabowski M., P. Kaniewski P., P. Serafin P.: Inertial navigation system for radar terrain imaging. IEEE/ION Position, Location and Navigation Symposium, 2016, pp.942-948.

[12] Konatowski S., Kaniewski P., Matuszewski J.: Comparison of Estimation Accuracy of EKF, UKF and PF Filters. Annual of Navigation 23/2016, pp. 69-87.

[13] Santana A.M., Aires K., Veras R., Medeiros A.: An Approach for 2D Visual Occupancy Grid Map Using Monocular Vision. Electronic Notes in Theoretical Computer Science, 281, 2014, pp. 175-191.

[14] Sola J., Marquez D., Codol J., Vidal-Calleja T.: An EKF-SLAM toolbox for MATLAB. 2009.

[15] Thrun S., Burgard W., Fox D.: Probabilistic Robotics. MIT Press, 2005.

[16] Thrun S., Montemerlo M., Koller D., Wegbreit B.: FastSLAM: An Efficient Solution to the Simultaneous Localization and Mapping Problem with Unknown Data Association. Proceedings of the AAAI-02 Conference on Artificial Intelligence, 2002, pp. 593-598.

[17] Vincent P., Rubin I.: A framework and analysis for cooperative search using UAV swarms. Proceedings of the 2004 ACM symposium on Applied computing, pp. 79-86.

[18] H. Wang H., F. Guixia F., L. Juan L., Y. Zheping Y., B. Xinqian B.: An Adaptive UKF Based SLAM Method for Unmanned Underwater Vehicle. Mathematical Problems in Engineering, 2013.

Received September 2018
Reviewed November 2018
Accepted December 2018

PIOTR KANIEWSKI
Military University of Technology
Urbanowicza 2, 00-908 Warszawa, Poland
e-mail: piotr.kaniewski@wat.edu.pl

PAWEŁ SŁOWAK
Military University of Technology
Urbanowicza 2, 00-908 Warszawa, Poland
e-mail: pawel.slowak@wat.edu.pl

STRESZCZENIE

W artykule przedstawiono problematykę i algorytm równoczesnego pozycjonowania i tworzenia mapy terenu (SLAM) przeznaczony dla bezzałogowych statków powietrznych (UAV). Opracowany przez autorów algorytm estymuje trajektorię lotu i tworzy mapę terenu znajdującego się pod UAV. Jako narzędzie estymacji położenia statku powietrznego zastosowano filtr cząsteczkowy. Zaproponowany algorytm poddano badaniom symulacyjnym. W artykule opisano opracowany przez autorów symulator przeznaczony do badania algorytmu SLAM oraz zamieszczono wybrane wyniki badań, zawierające utworzone mapy terenu i estymowane trajektorie UAV.