Multi-Agent SLAM Approaches for Low-Cost Platforms

Anton Filatov, Kirill Krinkin
Saint-Petersburg Electrotechnical University "LETI"
St. Petersburg, Russia
ant.filatov@gmail.com, kirill.krinkin@fruct.org

***Abstract*—Modern SLAM (Simultaneous Localization and Mapping) algorithms launched on a moving agent are bounded with its computation resources. The consistent way out is to add more computing agents that might explore the environment quicker than one and thus to decrease the load of each agent. This paper presents the state of art in area of Multi-agent SLAM algorithms and describes problems that are faced in front of a developer of such approach. The outstanding problem of Multi-agent SLAM – merging of maps built by separate agent during algorithm is also considered in this paper. Moreover the algorithm that extends laser 2D single hypothesis SLAM for multiple agents is introduced with evaluation of its performance.**

I. Introduction

When a mobile platform is placed in an unknown environment and it should simultaneously build a map and find its location, this problem is called SLAM problem. There are many algorithms that are applicable to a mobile moving platform, whether it is a robot vacuum cleaner, a reconnaissance drone, or even a rover. However, the process of building a map can be accelerated if several agents are used at once, where each explores its part of an environment, and in the future, the individual parts can be combined into one picture. The SLAM problem that is being solved by several agents is called Multi-agent SLAM problem below.

The first part of this paper continues the work [1] – presents brief state of art description of existing Multi-agent SLAM algorithms, describes most common approaches and extracts their advantages and disadvantages. It describes the approaches how to extend a single-agent SLAM algorithm to the Multi-agent case and clarifies which of them can be extended easier than other. It also demonstrates the approaches that were initially based on a Multi-agent architecture. Thus the high level description of state of the art approaches is presented in this paper in opposite to a previous work where the focus was to make a survey of several algorithms.

Secondly the question, that is commonly called map merging, about a combining results of several agents is discussed. Each existing algorithm solves this problem in its own way that is most commonly based on the architecture of current approach. The paper presents the high level state of art vision of this question and gives recommendation that are acceptable for different Multi-agent SLAM approaches.

The third part is a naive extension of a laser 2D grid based single-agent SLAM algorithm. The common idea of the suggested algorithm is to fulfill two statements:

- each agent should work individually, that means that there should not be a server that might make the whole system vulnerable to the lost of this agent;
- the algorithm should be successfully launched on the low performance hardware without huge delays and freezings.

To reach the first statement agents should contact to each other without an intermediary, i.e. agents should be gathered in for instance ad-hoc network. The second statement makes the restriction that the algorithm should base on simple approaches that leads to a trade off between the accuracy and the speed of performance.

The paper is structured as follows: in Section II there is the description of existing Multi-agent algorithms; Section III provides a description of the developed algorithm; its performance and testing on the real recorded datasets is presented in Section IV.

II. State Of Art

A. Roles for multi agents

There are several ways to classify Multi-agent SLAM algorithms (not considering the general division of SLAM algorithm such as visual or grid based, or single/multi hypothesis). In introduction it was said that there are two groups of algorithms: approaches that extend the Single-agent algorithm and those that are developed as Multi-agent initially. This divisions follows from the architecture of agents in the system: whether they all are independent and have equal rights, or there are two (or more) different roles: communication server, platform with sensors, computation unit etc.

In works [2], [3] the server-client architecture is presented. All agents that explore unknown environment only transfers data collected from sensors to server and do not perform any computations by themselves. The server should know the prior location of all agents to combine data from different agents in one map. If it doesn't have the prior knowledge than it should have several hypotheses about their locations to avoid wrong map merging.

The advantages of such system can be easily extracted:

- the system is resistant to the loss of client agents until all clients lose a connection to a server.
- client agents might be very cheap because their only purpose is to move, carry sensors and communicate to a server, and they should not process collected data.

ISSN 2305-7254

On the other hand the huge disadvantage that might overlap all pluses is the failure of the whole system in case of problems with server. If a server has problems with network connection or if it is down for any reason, agent by themselves will not be able to continue work or, even worse, they will work with no reason. Thus, approaches with independent agents such as [4], [5] or [6] become more popular despite they have their own problems described below.

B. Relative position of agents

Considering the architecture without a server, the first faced problem is a question about a relative orientation of agents. It could be whether known or unknown [7]. An algorithm that is not based on any assumption of agents initial position has more benefits in application to real life problems. Commonly there are three ways to calculate a relative orientation of agents:

- To make an assumption about exact initial agents position, track it through whole algorithm and thus have full information about positions of all agents while communication.
- To calculate the relative positions using on-board sensors such as video cameras. For example in [7] authors present the solution where they mark a moving platform with a red light that might be detected using omnidirectional camera.
- To superimpose foreign observations on existing map and calculate a position of other agents using, for example, a built-in scan matcher.

The last two ways are more preferable for autonomous agents but they add supervisor modules that should additionally track positions of other agents. In fact the choice of way often influences on the architecture of whole system.

C. Map merging

Another important problem for Multi-agent algorithms is so-called map merging – it is an approach for combining maps built by all agents in the system. The easiest solution comes from server-client architecture where only one copy of the map exists on server and clients update exactly this map. If the agents are independent and they communicate one to another then there might appear a conflict in maps built by independent agents and of the resolving of these conflicts is an open question. The conflict might appear if single SLAM algorithm failed and this failure was not detected properly. Commonly the solution of this problem differs from the type of SLAM algorithm. The probability methods are the most widespread and thus a naive method is to use average sum of maps but it leads to the situation when a broken map of one single agent breaks the map of all other agents.

Generally during map merging the following problems should be solved:

- What information should be shared by agents. Should it be the whole structure of a map (whatever it is) or only part of it.
- What should the agent do if the foreign map differs from its own.
- How much can the agent trust the parts in the foreign map that are unknown in own map.

High-level scheme of map merging is presented in Fig. 1.

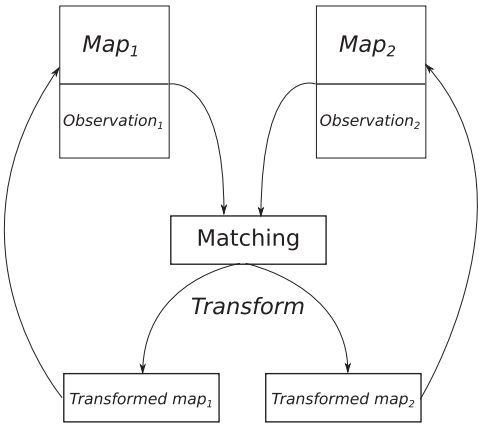


Fig. 1. Scheme of map merging

Some parts of this scheme might differ from one algorithm to another. For example in the most works [2], [7],[8] the return step between initial map and map with transform is trivial – a previous map is replaced with updated one. But in graph approaches the architecture allows to update a map, to detect collisions and drop repetitions in a map. This advantage decreases the performance speed and that is why in [4] instead of fair map merging only the transformation between maps is calculated.

Usually map merging process starts with calculation of relative position of agents and then combining maps. In [9] the openCV library was used for this purpose. Occupancy grid map was considered as an image to have an opportunity to extract ORB features and to find transformation between maps. This approach depends on the quality of a rangefinder and a map blur. The disadvantage of this approach appears when two maps completely differ both because of a failure of SLAM algorithm or the lack of common parts of maps.

Another problem of map merging is to find a consensus when two or more maps or parts of them differ one from other on separate agents. Consider an approach where the relative position of agents is determined correctly but the parts of built maps do not match. For graph based SLAM algorithm as it was mentioned above there are preset modules that can fix this mismatch, but for other approaches this question is more complex. In this case it is required to decide if it is necessary for $Transformed\ map_1$ to be equal to $Transformed\ map_2$ (see the Fig. 1). If they should, then some heuristics usually are applied or the final map consists only of that parts that coincide in both maps; if the final map differs on agents, then the merging algorithm can take into account the priority of maps, i.e. the own map may be considered as truthful and all conflicting parts on other maps may be dropped or merged with low probability. Also it is possible to calculate the correlation of maps and to detect (at least) conflicting parts.

The last question to be discussed is the appropriate time for connecting agents and transferring built maps. The most common approach is to start this process when two agents are

located in one place or close to each other. This condition allows to think that both agents captures the same observation and the map around meeting point is built identically. However in [5] there is the idea to split the observing space in non-crossing areas and to give each area to one agent. If an agent riches the area that belongs to another agent, it can communicate exactly to that agent and to get all the knowledge about full area that has been observed so far.

D. Typical extension of Single-agent SLAM algorithm

Below the tree types of laser grid based SLAM approaches are considered: feature based [7], multi-hypotheses approach as [8] and graph-based such as [4] and [5].

It is complicated to extend feature based SLAM to the Multi agent case. If features depend one from each other as in classic EKF SLAM, then the foreign covariation matrix should be merged with own matrix simultaneously fast and without both loss of data and duplication of features. This is the $O(n^2m^2)$ problem. If the features are independent as in FastSLAM then the question of map merging strictly depend on the knowledge of agents position. Authors of [7] suggest their solution for dropping duplications in covariance matrix after the agents calculate relative orientation using red bright marks placed on moving platforms.

In [8] one can find the approach to extend multi-hypotheses SLAM algorithm for multiple agents. In fact the main problem of this approach is the process of map merging. Authors come up with idea how to combine several hypotheses in one to send them to other agents. Commonly an agent should transfer all its hypotheses to allow its colleagues to choose the one that is the most suitable for their maps and moreover to update agent's own list of hypotheses costs. Authors of that paper suggest to send the average hypothesis to other agents and to calculate covariance for choosing the most suitable hypothesis of the foreign agent. Unfortunately the average approach often brings more noise then expected thats why the question of transferring multi hypothesis should be researched more accurate.

The easiest type of SLAM for extension on multi agents is a graph based SLAM, because the process of combining a foreign observation and an own map already exists in an implementation of graph SLAM and called loop closure. In Single-agent graph-based algorithm a new observation is inserted in a graph structure whether in new node or in existing one. This decision is based on a loop closure algorithm that detects if there are recurring nodes in a graph. This behavior fits the situation when it is necessary to insert a foreign observation in a graph. Such approach is implemented in [4]. However for graph based algorithm the map merging process becomes complex, because for example two maps of the same environment may contain different nodes (because different starting positions) and there is no obvious solution how to merge the different graphs without full rebuild. Authors of [4] have not implemented any map merging, they focused on calculating the transformation between agents.

III. MONTE-CARLO MULTI-AGENT SLAM

This section provides the description of proposed algorithm, its scheme and assumptions while quantitative results are provided in the next section. The algorithm is an extension of single-agent single-hypothesis grid based SLAM. This type was chosen to launch the algorithm on the a low performance hardware which might have not enough resources to handle multi-hypotheses or graph based algorithm. VinySLAM algorithm was selected as the core, being an accurate and fast representative of considered SLAM type [10].

In short vinySLAM algorithm is based on the Monte-Carlo scan matcher that calculates the current position with the observation and the map using the random walk. When the position is clarified it updates grid cells of map that represent Transferable Belief Model(TBM [11]). Thus except probability of being occupied a cell has the measure to be unknown or to be in a state of conflict. This model allows the single-hypothesis approach to be accurate enough to compete with multi-hypotheses or graph based architecture.

The advantage of TBM follows from the fact that the cells of grid map that are constructed according to this model have not only the probability of being occupied, but they contain quantified beliefs of these hypotheses: free, occupied, conflict, unknown. Thus if a cell in a map is free and according to a current observation this cell contains an obstacle, so the belief of "conflict" hypothesis of the cell increases instead of changing a probability of being occupied. The research [10] describes this model in details and shows the benefits of this approach according to SLAM problem and that is why vinySLAM is selected as the core for the proposed algorithm.

The communication architecture of proposed approach is presented in the Fig. 2.

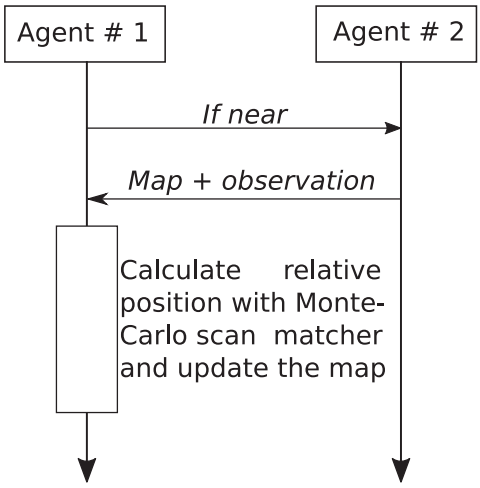


Fig. 2. Scheme of communication in proposed algorithm

The architecture allows to include any amount of agents in the system and all agents work independently and have equal roles. If one of agents falls down, all other would not recognize it and continue working. When an agent appears in neighborhood of another one, they start transferring maps and an observation. If they are located close one to another then it can be considered that they have visited (or they are visiting now) the same place and that means that the current foreign observation can be matched on the own built map.

The full process of data exchanging is following:

1) To detect a foreign agent that appears in the closeness of current position. This might be implemented using blue-tooth sensors or radio waves. The level of applicable closeness differs from environment and there exist a separate task to calculate the distance that would be a threshold. In this work an indoor environment was considered and the agents become close if the distance between them is less of equal than 1 meter.
2) To ask the foreign agent to provide its current observation and the occupancy map. As a map contains TBM cells and might be in resolution 1000x1000 cells, the amount of transferring data might be equal to several megabytes that means that the process of exchanging data might take up to 4-5 seconds.
3) To find the position of foreign agent using its observation. Basically the scan matching algorithm that is included in SLAM algorithm might be used for this purpose. As the foreign observation is located close to current position most of scan matching algorithms are applicable in this situation. In vinySLAM the Monte-Carlo scan matcher is used that finds the best position in a very close area to an initial position, thats why the distance of 1 meter can not be handled by this default scan matcher in appropriate time. In current work another version of Monte-Carlo scan matcher [12] was implemented.
4) When a relative position is calculated the map merging process begins. Basically the problem is to combine two 2D arrays of cells. As both agents perform the same vinySLAM algorithm the product maps have the same structure and there are points that are physically located in the same place. The combined map consists of both points that exist in two maps and unique points for each maps. If a point exists in both maps than there appears the question which occupancy would it have in the result map. If its probability of being occupied is equal for both maps (as it should be in perfect conditions) than the answer is obvious. But there might be a situation when these probabilities differ and the algorithm should combine, for instance, completely occupied and completely empty cell. In the proposed algorithm the TBM conjunction rule is applied for this situation.

The described algorithm fits both requirements that were presented in introduction: each agent might work individually and due to the approach of the core algorithm might be executed on a low performance hardware. The greatest restriction of such hardware is that the algorithm might have not enough time to finish accurate calculations and it should provide the result quickly. Monte-Carlo scan matching approach perfectly fits this statement because it can provide inaccurate result immediately and update the accuracy after a while. As more time it has as more precise result would be provided.

The disadvantage of the proposed algorithm is that the result of scan matching of foreign observation might fail, but there is no system that checks this. So if the relative position is calculated not accurate it leads to the wrong map merging, however in practice wrong map merging does not influence on the trajectory too much if the provided odometry is accurate enough.

IV. Performance

All tests were performed on the recorded data sequences from MIT [13] as they provide laser rangefinders observations and more or less accurate odometry. The problem of exploration of environment is not considered yet so testing on a recorded data is enough to clarify the accuracy of algorithm.

A. Multi agent data sequence converter

The recorded MIT dataset does not support multi agent interconnection, i.e. data files are structured in a way that does not allow to distinguish messages from two simultaneously played data sequences. Moreover all transmitted data pieces have unique timestamps that is stored as Unix-time so two different sequences have different Unix-time of start that does not allow to correlate these sequences without extra preparation.

To handle this problem the special tool was implemented. It suits for any dataset that is stored in the same format as MIT dataset (*.bag format [14]). The focus of this tool is to fix the timestamps of all messages and change the format to the relative time representation. In other words it sets up the beginning of data sequence to absolute zero. This means that the record time changes to 1970 year but it allows to take messages from different sequences simultaneously. The tool also is able to change names of topics that store messages to resolve the name conflicts.

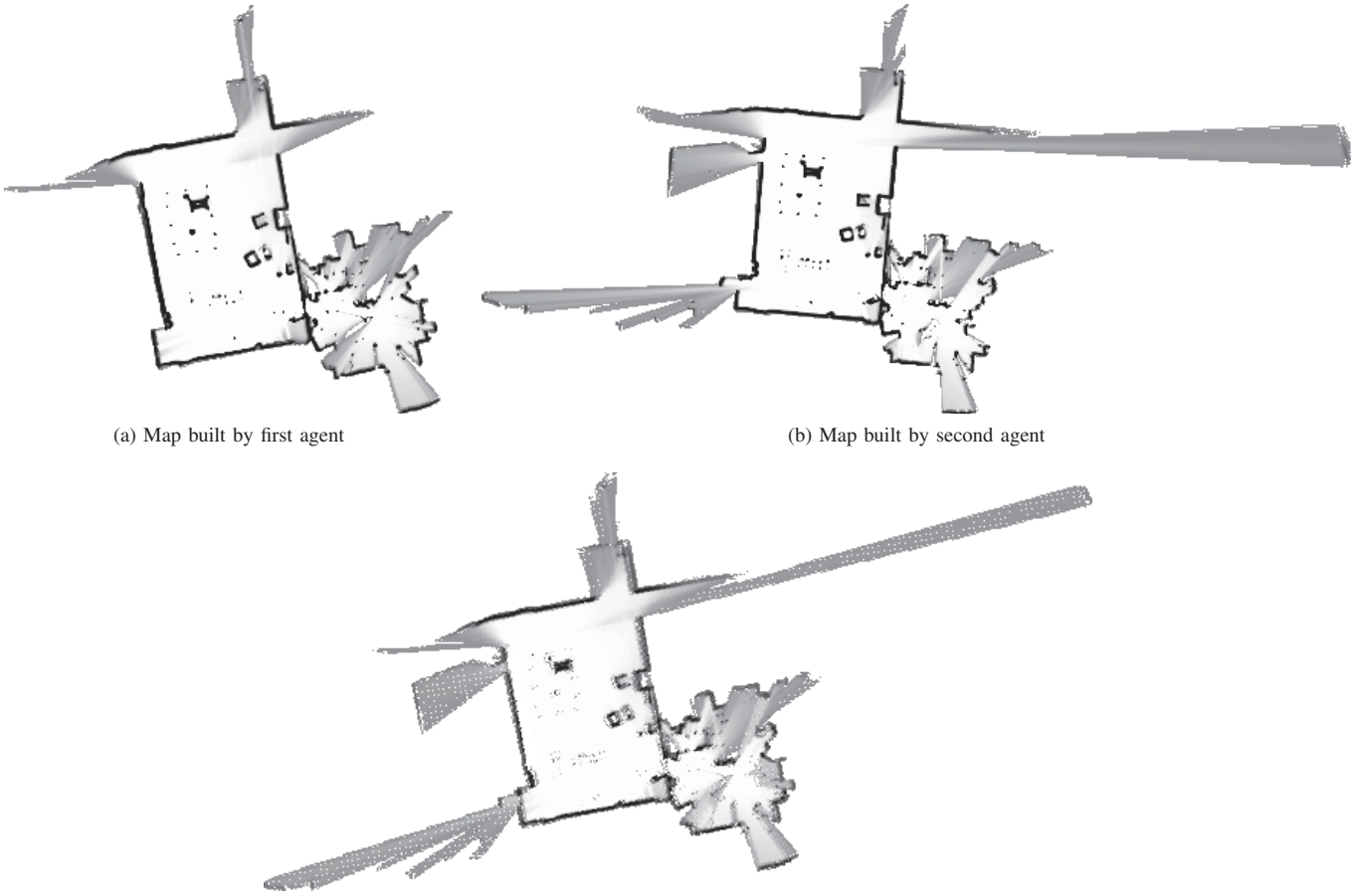
As this tool changes timestamps of messages, it can easily change the order of messages. For example it can make the data sequence to be played backwards or it can cut some parts of data sequence that allows to run any algorithm on a subsequence in both directions. This feature is helpful for testing considering algorithm because using this tool the most important parts of the real-world MIT data sequences can be segregated.

B. Evaluation of Multi-agent SLAM algorithm

To estimate the quality of Multi-agent system it is necessary to calculate how the existence of foreign agents effects the quality of built map and output trajectory of agent. As there are parts of the environment that are visited only by one agent, this agent should provide its map to others accurately and when they will visit this part of environment they will be able to solve only localization problem.

Therefore it is necessary to test proposed algorithm in the situation when two maps that are built by different agent have one common place and the rest of both maps differ. Another important test case is a real-life test when two agents explore environment independently and they exchange collected data through algorithm always when one reach another. Thus two situations for testing were considered:

- two different sequences are played simultaneously that provides the situation when two agents explore environment independently and when they meet, they update some parts of their maps and add new cells in their map as it is shown in the Fig. 3. To estimate the quality of algorithm beside a map the output trajectory for both agents is compared to the ground-truth.



(a) Map built by first agent

(b) Map built by second agent

(c) Combined map

Fig. 3. Combining two different sequences

The root mean square error (RMSE) is calculated to represent the difference between trajectories;

- two versions of one sequence are played simultaneously, one of copy is played forward and another backward, and in the middle of the path agents perform map merging. The second part of the path each agent solves the localization problem and in perfect conditions does not update its map. This case represents the situation when two agents explore different parts of environment and at the end they combine captured results. Quantitative estimation is also performed by comparison the output with ground-truth. If a map merging process failed than the output trajectory would differ from ground-truth and this would be presented in RMSE. The example of this situation is presented in the Fig. 4.

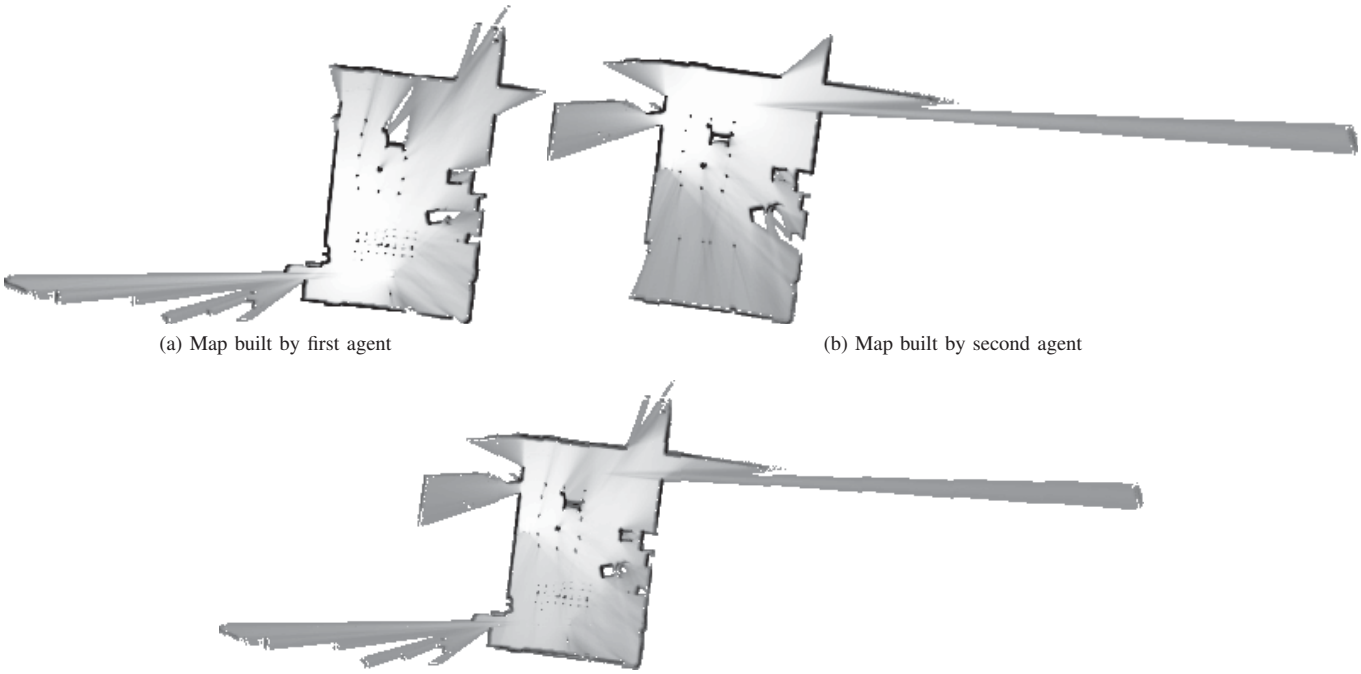
In both cases the trajectory of agents movement is the part or full ground-truth trajectory and this means that it is possible to calculate how much each agent deviated from the real trajectory after map merging and therefore it is possible to estimate how accurate the map merging was performed. First case presents a regular situation when agents explore the environment and accidentally reaches each other, when the second case stresses the situation when an agent moves to the unvisited environment but have a map that is built by another agent.

Tests of accuracy were performed on computation unit with following characteristics. Intel® Core™ i7-860 4x2.8GHz with DDR3 8GB, Ubuntu Xenial x64. The results for several sequences for case 1 from above are presented in the Table I. The results of second case are put in the Table II.

TABLE I. RMSE VALUES FOR DIFFERENT SEQUENCES

The sequence	Dist, m	RMSE, m
2011-01-20-07-18-45	76	0.045 $\pm$ 0.005
2011-01-21-09-01-36	87	0.080 $\pm$ 0.018
2011-01-24-06-18-27	87	0.096 $\pm$ 0.007
2011-01-25-06-29-26	109	0.094 $\pm$ 0.006
2011-01-28-06-37-23	145	0.361 $\pm$ 0.175
2011-01-27-07-49-54	94	0.170 $\pm$ 0.019

The results show the accuracy of proposed algorithm, as the output error is always not greater than 0.5m that is applicable for such indoor environment as MIT. To be honest this result is mostly based on the accuracy of single-agent version of vinySLAM algorithm. The greatest troubles might appear if map merging brought a little error in rotation, because despite in this case the output map would be consistent, the output trajectory might noticeably differ. Though this problem appears in any Multi-agent algorithm and even graph based



(a) Map built by first agent

(b) Map built by second agent

(c) Combined map

Fig. 4. Combining two subsequences

TABLE II. RMSE VALUES FOR SPLIT SEQUENCES

The sequence	Dist, m	'forward' agent RMSE, m	'backward' agent RMSE, m
2011-01-20-07-18-45	38 + 24	0.044 $\pm$ 0.015	0.021 $\pm$ 0.005
2011-01-21-09-01-36	43 + 31	0.063 $\pm$ 0.011	0.068 $\pm$ 0.012
2011-01-24-06-18-27	43 + 41	0.086 $\pm$ 0.009	0.071 $\pm$ 0.003
2011-01-25-06-29-26	33 + 61	0.033 $\pm$ 0.013	0.097 $\pm$ 0.016
2011-01-28-06-37-23 part 1	41 + 39	0.184 $\pm$ 0.093	0.099 $\pm$ 0.040
2011-01-28-06-37-23 part 2	41 + 39	0.311 $\pm$ 0.187	0.272 $\pm$ 0.195
2011-01-27-07-49-54	31 + 62	0.142 $\pm$ 0.019	0.161 $\pm$ 0.022

approach can fix it only in specific conditions, when agents visit controversial part of environment several times.

After the quality of proposed algorithm is discussed it is necessary to find out its performance. For this case the algorithm was launched in Raspberry Pi model 3B. The mean scan matching time for considered sequences was 102.1 ms, so for performing this algorithm in real time it can take about 10 scans per second. The map merging process took 7.4 seconds and that means that this algorithm requires agents to stay in one place while exchanging data. This speed might be increased if a map data would be compressed.

V. CONCLUSION

The paper presents the state of art for modern Multi-agent SLAM algorithms and the scheme that describes most common approaches. To sum up a graph based algorithms become the most popular in multi-agent research area. This may be explained by the fact that the algorithm of inserting foreign observations into own map already exist in Single-agent graph based algorithm and moreover such algorithm can completely update the map if the foreign observations differ from built map. On the other hand graph based algorithm can not be launched in real time on low performance devices and thus it is required to look for alternative approaches.

Moreover the problem of map merging is described in this paper. Map merging approach strictly depends on the type of core SLAM algorithm and on the architecture of Multi-agent system. For graph based approaches the process of map rebuilding using new observations is already built in single-agent SLAM and it can be easily applied for map merging. However for other types of Multi-agent SLAM algorithms the problem of consensus is more complex and naive average calculations may break an output result.

Finally the non-graph single hypothesis multi-agent SLAM algorithm was proposed. The advantage of this algorithm is that it can be launched on low performance devices and its accuracy is well that was achieved using fast and accurate core algorithm. The map merging process is implemented using conjunction from Transferable belief model that allows to detect conflicts between maps and updates a map using only valid data.

Plans for the future work are to update the map merging rule according a location of conflict cells. TBM structure of cells allows to find clusters of conflict cells and this knowledge may help to determine which areas should be explored more careful. Also in the future it is required to increase the speed of map merging so that it could be used in the real time. This might be achieved after solving two problems: to increase the performance of scan matcher that should correctly find another

agent in a distance of several meters, and to compress the transferred map that consists of TBM cells which contain four probabilities.

VI. Acknowledgment

Authors would like to thank JetBrains for provided support and materials for working on this research.

References

[1] K. Krinkin, A. Filatov, and A. Filatov, "Modern multi-agent slam approaches survey," in *Proceedings of the XXth Conference of Open Innovations Association FRUCT*, vol. 776, pp. 617–623, Directory of Open Access Journals, 2017.

[2] C. Forster, S. Lynen, L. Kneip, and D. Scaramuzza, "Collaborative monocular slam with multiple micro aerial vehicles," in *2013 IEEE/RSJ International Conference On Intelligent Robots And Systems (IROS)*, no. EPFL-CONF-199731, pp. 3963–3970, 2013.

[3] D. A. Castro, C. F. Morales, and R. F. De la Rosa, "Multi-robot slam on client-server architecture," in *Robotics Symposium and Latin American Robotics Symposium (SBR-LARS), 2012 Brazilian*, pp. 196–201, IEEE, 2012.

[4] M. T. Lazaro, L. M. Paz, P. Pinies, J. A. Castellanos, and G. Grisetti, "Multi-robot slam using condensed measurements," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pp. 1069–1076, IEEE, 2013.

[5] T. Cieslewski, S. Choudhary, and D. Scaramuzza, "Data-efficient decentralized visual slam," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2466–2473, IEEE, 2018.

[6] S. Urban and S. Hinz, "MultiCol-SLAM - a modular real-time multi-camera slam system," *arXiv preprint arXiv:1610.07336*, 2016.

[7] X. S. Zhou and S. I. Roumeliotis, "Multi-robot slam with unknown initial correspondence: The robot rendezvous case," in *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pp. 1785–1792, IEEE, 2006.

[8] N. E. Özkucur and H. L. Akın, "Cooperative multi-robot map merging using fast-slam," in *Robot Soccer World Cup*, pp. 449–460, Springer, 2009.

[9] L. Jian, Z. Chen, I. Qiang, W. Heng, and M. Manzhen, "Vision feature extraction algorithm for occupancy grid maps merging," in *Proceedings of the 2017 2nd International Conference on Communication and Information Systems*, pp. 290–293, ACM, 2017.

[10] A. Huletski, D. Kartashov, and K. Krinkin, "Vinyslam: an indoor slam method for low-cost platforms based on the transferable belief model," in *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*, pp. 6770–6776, IEEE, 2017.

[11] P. Smets and R. Kennes, "The transferable belief model," *Artificial intelligence*, vol. 66, no. 2, pp. 191–234, 1994.

[12] D. Fox, W. Burgard, F. Dellaert, and S. Thrun, "Monte carlo localization: Efficient position estimation for mobile robots," *AAAI/IAAI*, vol. 1999, no. 343-349, pp. 2–2, 1999.

[13] M. Fallon, H. Johannsson, M. Kaess, and J. J. Leonard, "The mit stata center dataset," *The International Journal of Robotics Research*, vol. 32, no. 14, pp. 1695–1699, 2013.

[14] "Ros bags." http://wiki.ros.org/Bags.