Abstract: Sensor-based coverage problems have many applications such as patrolling, search-rescue, and surveillance. Using multi-robot team increases efficiency by reducing completion time of a sensor-based coverage task. Robustness to robot failures is another advantage of using multiple robots for coverage. Although there are many works to increase the efficiency of coverage methods, there are few works related to robot failures in the literature. In this paper, fault-tolerant control architecture is proposed for sensor-based coverage. Robot failures are detected using the heartbeat strategy. To show the effectiveness of the proposed approach, experiments are conducted using P3-DX mobile robots both in laboratory and simulation environment.

Keywords: Fault-tolerant, multi-robot, sensor-based coverage, control architecture

1. Introduction

Multi-robot sensor-based coverage path planning problem requires that every point in a given area should be covered by at least one member of the robot team using its sensors (Acar et al., 2006). This problem appears in applications such as landmine detection, patrolling, search-rescue, and surveillance (Trevai et al., 2007).

In the literature, there are several studies on multi-robot coverage problem. In an early work (Kurabayashi et al., 1996), the configuration space and Voronoi diagram are used to compute paths in the whole area. Firstly, a tour is generated for travelling all the paths, and then appropriate parts of the tour are assigned to each robot according to the cost evaluation. The cost is evaluated in terms of the traveled distance by each robot. The approach in (Latimer et al., 2002) presents an adaptation of the single-robot cellular decomposition approach to multi-robot teams. A robot team moves in a formation to cover cells. If obstacles divide the environment into sub-regions, team splits up into smaller teams to continue coverage task. In (Kong et al., 2006), authors build their approach on a single-robot coverage algorithm, boustrophedon decomposition. The robots are initially distributed through the free space. Then, each robot is allocated a virtually bounded area to cover. The area is decomposed into cells with fixed width. In (Mei et al., 2006), mobile robot deployment problem is considered for a specific type of coverage problem. The deployment problem is described as determining the number of groups to be unloaded by a carrier, the number of robots in each group, and the initial locations of those robots. Both timing and energy constraints of robots are considered. Power consumption of mobile robots is modeled and the energy cost of possible scanning lines for coverage are calculated. Then, the deployment problem for the members of the team is solved for each robot considering their energy capacities. In (Parlaktuna et al., 2009), an approach based on capacitated arc routing problem (CARP) is proposed and applied to multi-robot sensor-based coverage planning for narrow interior environments. The proposed algorithm is developed by modifying Ulusoy's partitioning algorithm. Basically, the proposed algorithm uses CPP/RPP (Chinese Postman Problem/Rural Postman Problem) solving techniques in the initial phase; then it partitions the tour among the robots considering their energy capacities.

These works intend to increase the efficiency of multi-robot coverage problem; but, they do not consider robot failures. If one of the robots fails, the regions assigned to this robot must be covered by other robots to guarantee complete coverage of the given area. Therefore, multi-robot control architectures must be designed to handle such failures. If the robot is not able to inform the team about its failure, the team must detect the failure and distribute the uncompleted tasks of the failed robot among the remaining robots. The work by (Hazon, N. & Kaminka, G.A., 2008) is one of the few works that considers robot failures for multi-robot coverage. The authors analytically showed that their proposed algorithms are robust in that, as long as a single robot is able to move, the coverage will be completed. This work is proposed in algorithmic level; therefore, robot failure detection and software architectures are not given.

In this study, fault-tolerant control architecture is proposed for sensor-based coverage. The paper is
organized as follows: In Section 2, fault-tolerant control architecture is explained. Applications of the proposed approach are given in Section 3. Conclusions and discussions are given in Section 4.

2. Fault-Tolerant Control Architecture

The proposed fault-tolerant control architecture consists of software agents performing specific functions like communication with other robots, perception, action, fault detection, planning, human interaction, etc. Fig. 1 shows overall block diagram of the architecture for \( n \) robots. Each dashed-square shows a robot in the team, and each block inside a robot represents a software agent. In the proposed architecture, each robot has four functional software agents: user interface agent (UIA), planner agent (PA), action agent (AA), and communication agent (CA). The functional details of the agents are defined as follows:

**User Interface Agent (UIA):** The main task of this agent is to provide interaction between the robot and users. It provides a user-friendly environment to define tasks and provides information about the internal state of the robot via terminal or graphical interface.

**Planner Agent (PA):** This agent is responsible for generating plans which may include motion planning, path planning, task planning, etc. It contains planning mechanisms which use the planning algorithms for different task domains. It is able to generate plans for one or more robots.

**Action Agent (AA):** This agent is responsible for tasks related to hardware of robots (actuators and sensors). It performs behavior-based motions according to motion plans provided by the PA. The task-oriented behaviors are implemented in conjunction with the survival behaviors, such as obstacle avoidance, by coordination of subsumption approach (Murphy, R.R., 2000). This agent is also responsible for robot’s high-level perception and localization.

**Communication Agent (CA):** This agent is responsible for interaction with the other robots. The interaction takes place over the infrastructure which is realized using TCP/IP. The robots communicate each other explicitly via the CA. This agent may also be responsible for some other tasks like fault detection as in our study. All of the agents are in the separate computational processes which perform tasks according to their designed purposes. The messaging between the agents is managed by another software agent, facilitator. Open Agent Architecture (OAA) (Martin et al., 1999) is used as an infrastructure framework to provide content-based message transfer between the agents.

Planning and fault detection play important roles in the proposed architecture. In the following sub-sections, planning for coverage problem and the fault detection mechanism are explained in more details.
Step 2. Construct an Euler tour that covers all the required edges using the RPP approach.

Step 3. Partition the Euler tour using Dynamic Ulusoy Algorithm (Yazici, et al., 2009).

In the last phase, the partial plans are dispatched to each robot via the CA while its own plan is sent to the AA.

2.2. Fault Detection

Robot failures are classified into three fundamental categories: (i) Communication failures, (ii) partial robot malfunctions, and (iii) robot death (Dias et al., 2004). Communication failures take place in distinct domains and ranges from occasional loss of messages to loss of all communication. If a partial malfunction occurs in one of the abilities of the robot, the robot loses its ability to effectively use some of its resources but retains the ability to use other resources. If this malfunction is related to the assigned task, this robot may be considered dead. In the final category, the robot may die as a result of battery depletion, a hardware breakdown, or the crash of the software operating system. In all of these cases, robot failure must be detected by a detection mechanism.

Several methods were proposed to deal with the fault detection problem. One of the classical techniques of failure detection is socially attentive monitoring which is based upon detecting the failed robot by its teammates (Parker, L., 1998). Other approaches are pinging and heartbeat strategies (Bertier et al., 2002). In the pinging strategy, an agent periodically sends requests to the other agents and waits their replies (Christensen, A.L., 2009). However, in the heartbeat strategy the agent periodically sends heartbeat messages known as “I’m alive” to other agents in order to inform them about its aliveness. At first, the heartbeat strategy is generally applied on processes in multi-agent systems (Faci, N., 2006; Pasin, M., 2008) and then it is applied to mapping and target acquisition tasks in multi-robot systems (Barnhard, D., 2005; Dobre, C. M., 2009).

In this study, the heartbeat strategy is used to detect the failed robots. The heartbeat strategy system for an n-robot team (\(\Pi = \{MR_1, MR_2, ..., MR_n\}\)) is implemented by sending and receiving messages via the CA. Message traffic between a pair of robots (\(MR_i\) and \(MR_j\)) is shown in Fig. 4.

In this figure, the heartbeat period of \(MR_i\), \(\Delta_i\), is the time between two “I’m alive” messages. \(t_r\) is the time at which the last heartbeat message from \(MR_i\) is received by \(MR_j\). The control time, \(t_{control-j}\), is used to periodically check whether the heartbeat signal from \(MR_i\) is received or not by \(MR_j\). The delay time, \(\Delta_{td-i}\), is the difference between the control time and the time of the last received heartbeat message (\(\Delta_{td-i} = t_{control-j} - t_r\) where \(t_{control-j} \geq t_r\)). The timeout, \(\Delta_{to-i}\), is a predefined duration which should be greater than \(\Delta_i\). \(MR_j\) concludes that \(MR_i\) has a failure when \(\Delta_{to-i} < \Delta_{td-i}\).
Fig. 4. Heartbeat strategy for mobile robots

If $\Delta_{\text{to}} - \Delta t$ is short, the failures are detected in a short time, but likelihood of false positive failures increases due to communication delays. If the maximum communication delay, $\Delta t_{\text{cd, max}}$ in the network is known, $\Delta_{\text{to}}$ should be greater than $\Delta t + \Delta t_{\text{cd, max}}$.

This heartbeat strategy is implemented using the control architecture as explained in the following subsection. Since each robot is able to detect the failure of any other robot in the team, this is a distributed fault detection strategy.

2.3. Fault detection mechanism in the proposed architecture

Fault-tolerant coverage planning is achieved by implementing the heartbeat strategy in agent-based robot control architecture. The fault detection mechanism explained in the previous section uses a status table stored on each robot. The structure of the table is given in Table 1. This table holds information about other robots in the team.

In this table, the first and the second columns hold unique numbers and planning priorities of other robots, respectively. The third column represents the last received heartbeat message’s time from $MR_j$ ($j = 1, \ldots, n, j \neq i$). The fourth column holds the planning indexes of other robots. The planning index is used to determine approximate physical location of the robot when the last heartbeat signal is received. The last column holds the fault state of the robot $i$, $FS_i$, as alive or dead. This status table is handled and updated by the $CA_i$ of each robot.

Table 1. Status table for $MR_i$

| Robot ID | Priority | Last Heartbeat Time | Planning Index | Fault State |
|----------|----------|---------------------|----------------|-------------|
| $MR_1$  | $P_1$    | $t_{r1}$            | $G_1$          | $FS_1$      |
| $MR_2$  | $P_2$    | $t_{r2}$            | $G_2$          | $FS_2$      |
| $\ldots$| $\ldots$| $\ldots$           | $\ldots$      | $\ldots$    |
| $MR_{r-1}$ | $P_{r-1}$ | $t_{r-1}$        | $G_{r-1}$      | $FS_{r-1}$  |
| $MR_{r+1}$ | $P_{r+1}$ | $t_{r+1}$        | $G_{r+1}$      | $FS_{r+1}$  |
| $\ldots$| $\ldots$| $\ldots$           | $\ldots$      | $\ldots$    |
| $MR_n$  | $P_n$    | $t_{rn}$            | $G_n$          | $FS_n$      |

The sequence diagram of the fault detection process performed by the $CA_i$ is given in Fig. 5. In this sequence diagram, the $CA_i$ realizes three tasks as explained below.

1. **Message sending**: In this phase, each robot prepares a status message and sends it periodically to all other robots at the end of the predetermined heartbeat time interval. This message consists of robot’s id and its plan index value (“status(robotID, plan_index)”). This shows that the robot is alive on that instant.

2. **Message receiving**: When a robot receives status message from any robot, it updates the corresponding row of the table (Table 1). The last heartbeat time entry of the table is updated according to the local time of the receiving robot.

3. **Status Check**: All robots check their tables periodically at the end of $\Delta_{\text{control}}$ duration. On control time, $t_{\text{control}_i}$, the robot compares its local time with the last heartbeat time of all other robots. If the time interval between the last heartbeat time and the control time is greater than $\Delta_{\text{to}_i}$, then the robot $i$ is considered to be dead.

If robot failures are detected, then the ID of the failed robots are sent to all other robots with a message of `robotFailure('failedID1, failedID2, …')`. Then, the $CA$ of the robot with the highest priority informs its $PA$ about the failure with a message of `planFaultDetection("failedID1, failedID2, …", "plan_index1, plan_index2, …")`. Finally, the $PA$ gathers planning indexes of alive robots to initiate planning as in Fig. 3.

Fig. 5. Sequence diagram for the fault detection
3. Application

The proposed fault-tolerant system is coded in C++ and tested using P3-DX robots both in laboratory and MobileSim simulation environments.

3.1. Application in Laboratory Environment

A platform at ESOGU Artificial Intelligence & Robotic Laboratory (AIRLAB, 2009) is used as the test bed (Fig. 6). In the applications, P3-DX robots are used. The P3-DX robot has an onboard P3-800 computer with Linux OS. The sensors on the robot are: the SICK LMS laser range finder, sonar sensors, camera, and compass. A wireless network is set for communication among robots and computers. Mainly the SICK LMS laser range sensor is used for the coverage task. This sensor has normally a range of 50 meters, but for experimental purposes the range is restricted to 3 meters with software. A topological map of the platform and GVD-based network is set for communication among robots and computers. In applications, localization is realized by using the ARNL software module (ARNL, 2007).

In the first experiment, two P3-DX mobile robots (MR1 and MR2), initially at node 1, are required to cover the given environment. After the coverage task was entered by the user via the UIA1, the PAi generated the paths for both robots. The tours are as follows; the tour for MR1: (1-3), (3-4), (4-5), (5-6), (6-7), (7-8), (8-9), (9-10), (10-11), (11-8), (8-14), (14-15), (15-16), (16-13), (13-14), (14-13), (13-12), (12-7), (7-6), (6-2), (2-1) and the tour for MR2: (1-2), (2-6), (6-7), (7-12), (12-17), (17-18), (18-19), (19-20), (20-17), (17-12), (12-7), (7-6), (6-2), (2-1). The robots cover the bold-written edges and passes through the italic-written edges without coverage. The robots cover the given area as shown in Fig. 8 with no failure. Fig. 8 is drawn using the logged location values of the robots during the experiment.

Another experiment is conducted to test the proposed fault detection mechanism. Robots start this experiment by using the same plan of the previous experiment. Up to 120 seconds, the experiment continued as in the previous experiment. At $t=120\text{sec}$, MR1 is manually turned off. The fault detection mechanism and replanning results are explained in the following paragraphs.

The parameters of the fault detection mechanism are:

- $\Delta_1 = \Delta_2 = 10\text{sec}$,
- $\Delta_{\text{control-1}} = \Delta_{\text{control-2}} = 15\text{sec}$, and
- $\Delta_{\mu-1} = \Delta_{\mu-2} = 20\text{sec}$.

Each robot logs its received and sent messages during the experiment for a detailed fault detection analysis.

Fig. 9 shows the heartbeat messages of MR1 received by MR2. As shown in this figure, MR2 regularly receives the heartbeat messages of MR1 up to 120 seconds, and controls these messages from MR1 at the end of each control period of MR1.

In Fig. 10, the heartbeat signals of MR2 received by MR1 are given. Up to 120 seconds, the heartbeat signals are received normally. The last heartbeat message from MR2 has been received at $t=113\text{sec}$. The next control time is $t=120\text{sec}$. At $t=120\text{sec}$, the delay time for the heartbeat signal of MR1, $\Delta_{\mu-2}$, is 7 seconds ($t_{\text{control-1}} = t_{\text{control-2}} = 120\text{sec}$, which is smaller than $\Delta_{\mu-2}$). Thus, MR1 does not change the status of MR2. Since MR2 is killed manually at $t=120\text{sec}$, after that instant, it is not able to send any heartbeat message to MR1. At the next control time, $t=135\text{sec}$, of MR1 the new value of $\Delta_{\mu-2}$ is 22 seconds, which is greater than $\Delta_{\mu-2}$. Thus, MR1 changes the status of MR2 from alive to dead. Note that MR2 is between nodes 14 and 15 when it detects the failure of MR1 which is turned off between nodes 19 and 20. Due to this failure, MR2 is not able to cover the edges (20-17) and (12-7). Fig. 6 shows approximate locations of the robots at that instant.
3.2. Fault detection test in simulation environment

The response of the proposed fault detection mechanism is tested in MobileSim simulation environment. The same fault detection parameters of the previous experiments are used. The simulation is started with five robots, and one robot is killed after every 50 seconds. Fig. 12 shows the status of robot aliveness versus time graph. The thick lines show the duration at which the robot remains alive. Thin lines show the time period between the robot failure and the detection of this failure by the proposed fault detection mechanism. Dashed lines indicate the remaining time period.

According to the thin lines, the failures of MR1, MR2, MR3, and MR4 are detected 25, 20, 15, and 25 seconds later after their failures, respectively.

Moreover, the maximum communications delay time ($\Delta_{cd\text{, max}}$) is recorded as 5 seconds. Therefore, a timeout value of 20 seconds is acceptable for five robots considering the maximum communication delay time. A more detailed analysis related to communication delay vs number of robots is given in the following section.

3.3. Analysis of communication delays for higher number of robots

The communication delay is analyzed up to 25 robots using the proposed control architecture. The columns of Table 2 show the average, the minimum, and the maximum delay for each experimental setting. Heartbeat signals are sent every 10 seconds, the control time is 15 seconds and the fault detection time is set to 200 seconds. As seen in Table 2, the maximum communication delay increases polynomially as the number of robots increases.

| Number of Robots | Average Delay | Minimum Delay | Maximum Delay |
|------------------|--------------|--------------|---------------|
| 5                | 3.69         | 0            | 9             |
| 10               | 8.14         | 0            | 24            |
| 15               | 23.78        | 1            | 98            |
| 20               | 48.82        | 2            | 132           |
| 25               | 91.45        | 21           | 188           |

Table 2. Communication delays
To avoid false-positive fault detections, the minimum timeout value should be selected higher than the sum of the maximum communication delay time for the given number of robots and the heartbeat period.

4. Conclusion

In this study, fault-tolerant control architecture is proposed and tested for sensor-based coverage problem by using P3-DX robots both in laboratory and MobileSim simulation environments. Robot failures are detected using the heartbeat strategy that does not require time synchronization between robots. Since the failure can be detected by any robot in the team, the proposed control architecture is distributed in terms of fault detection.

Communication delay time changes proportional to the number of robots; therefore, timeout value should be selected carefully according to the number of robots in the team.

The proposed control architecture has a modular structure. In this work, for sensor-based coverage problem, CARP-based planning algorithm is used in the PA of the architecture. Other types of planning algorithms may also be used for planning. Besides, other multi-robot missions such as object handling and transportation, serving patients and elderly in hospitals, etc. can easily be implemented by using the proposed robot control architecture.

In this study, it is assumed that the communication network is fully connected. This assumption results in a polynomial increase of communication delay as the number of robots increases. Therefore, a wise communication strategy to reduce the time delay should be used. For instance, one robot may only communicate with the nearest partner while the whole communication network is kept to be connected. In the future studies, the authors plan to search the literature for communication strategy to reduce the communication delays.

5. Acknowledgement

This work is supported by the Scientific and Technical Research Council of Turkey (TUBITAK), Contract No 107E064, project title: “Mobile Robot Route Planning for Complete Coverage of Dynamic Indoor Environments”.

6. References

Acar, E.U., Choset H. & Lee, J.Y. (2006). Sensor-based coverage with extended range detectors, IEEE Transactions on Robotics, vol. 22 (1), pp. 189-198.

Ahmadabadi, M.N. & Ghaderi, F. (2004). Distributed Cooperative Load Redistribution for Fault Tolerance in a Team of Four Object-Lifting Robots, Advanced Robotics, Vol. 18, No.1, pp. 61-81

AIRLAB, Artificial Intelligence & Robotics Laboratory, Eskisehir Osmangazi University (2009). Available from: http://www.ai-robotlab.ogu.edu.tr, Accessed: 2009-05-15

Arai, T., Pagello, E. & Parker, L.E. (2002). Advances in Multi-Robot Systems, IEEE Transactions on Robotics and Automation, 18 (5), pp. 655-661

ARNL Installation and Operations manual (2007), Version 2.1, Available from: http://robots.mobilerobots.com/, Accessed: 2009-05-15

Barnhard, D. (2005). Distributed Collaborative Robotic Mapping, Ms. Thesis, B.A. Cognitive Science, University of Georgia

Bertier, M., Marin, O. & Sens, P. (2002). Implementation and performance evaluation of an adaptable failure detector, IEEE International Conference on Dependable Systems and Networks (DSN 2002), pp. 354 – 363

Christensen, A.L., O’Grady, R. & Dorigo, M. (2009). From Fireflies to Fault-Tolerant Swarms of Robots, IEEE Transactions on Evolutionary Computation, Vol. 13, No. 4, pp. 754-766.

 Dias, B., Zinck M., Zlot, M. & Stentz, A. (2004). Robust Multirobot Coordination in Dynamic Environments, IEEE International Conference on Robotics and Automation (ICRA ‘04), Vol.4, pp. 3435-3442, 0-7803-8232-3.

Dobre, C. M., Pop, F., Costan, A., Andreica, M. I. & Cristea, V. (2009). Robust Failure Detection Architecture for Large Scale Distributed Systems, Proc. of the 17th Intl. Conf. on Control Systems and Computer Science (CSCS), vol. 1, Bucharest, Romania, pp. 433-440.

Faci, N., Guessoum, Z. & Marin, O. (2006). DimaX: a fault-tolerant multi-agent platform, Proceedings of the 2006 international workshop on Software engineering for large-scale multi-agent systems, Shanghai, China, pp. 13 – 20.

Gerkey, B.P. & Mataric, M.J. (2002). Sold!: Auction Methods for Multirobot Coordination, IEEE Transactions on Robotics and Automation, Vol.18, No.5, pp. 758 – 768.

Hazon, N. & Kaminka, G.A. (2008). On redundancy, efficiency, and robustness in coverage for multiple robots, Robotics and Autonomous Systems, Vol. 56, No. 12, pp. 1102-1114.

Kong, C.S., New, A.P. & Rekleitis, I. (2006). Distributed Coverage with Multi-Robot System, Proceedings of 2006 IEEE International Conference on Robotics and Automation, pp. 2423-2429.

Kurabayashi, D., Ota, J., Arai, T. & Yoshida E. (1996). Cooperative sweeping by multiple mobile robots, Proc. IEEE International Conference on Robotics and Automation, vol. 3, pp. 1744-1749.

Latimer, D., IV Srinivasa, S., Lee-Shue, V., Sonne, S., Choset, H. & Hurst, A. (2002). Towards sensor based coverage with robot teams, Robotics and Automation
Proceedings (ICRA '02), vol. 1, pp. 961-967, 0-7803-7272-7.
Long, M., Murphy, R.R. & Parker, L.E. (2003). Distributed multi-agent diagnosis and recovery from sensor failures, In Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 2506–2513.
Martin, D., Cheyer, A.J. & Moran, D.B. (1999). The Open Agent Architecture: A Framework for Building Distributed Software Systems, Applied Artificial Intelligence, vol. 13, pp. 91-128.
Mei, Y., Lu, Y.H., Hu, Y.C. & Lee, C.S.G. (2006). Deployment of mobile robots with energy and timing constraints, IEEE Transactions on Robotics and Automation, Volume 22, Issue 3, pp. 507-522.
Murphy, R.R. (2000). Introduction to AI Robotics, MIT Press, 0-262-13383-0.
Parker, L. (1998). ALLIANCE: An Architecture for Fault Tolerant Multirobot Cooperation, IEEE Transactions on Robotics and Automation, Vol.14, No.2, pp. 220 – 240.
Parlaktuna, O., Sipahioglu, A., Kirlik, G. & Yazici, A. (2009). Multi-robot sensor-based coverage path planning using capacitated arc routing approach, the 24th IEEE International Symposium on Intelligent Control (ISIC), Saint Petersburg, Russia, pp.1146-1151.
Pasin, M., Fontaine, S. & Bouchenak, S. (2008). Failure Detection in Large Scale Systems: a Survey, IEEE Network Operations and Management Symposium Workshops, 2008, pp. 165-168.
Trevai, C., Ota, J. & Arai, T. (2007). Multiple mobile robot surveillance in unknown environments, Advanced Robotics, vol. 21, pp. 729-749.
Yazici, A, Kirlik, G., Parlaktuna, O. & Sipahioglu, A., . (2009). A Dynamic Path Planning Approach for Multi-Robot Sensor-Based Coverage Considering Energy Constraints, The 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems , St. Louis, MO, USA, pp. 5930-5935.