Hybrid Multi-Robot System for Drilling and Blasting Automation

Dac Dang Khoa Nguyen, Yujun Lai, Sheila Sutjipto and Gavin Paul

***Abstract*— Multi-robot systems possess the potential of becoming the next generation of robots in the mining industry due to their robustness and scalability. However, they present challenges for the system to efficiently allocate tasks to each robot and allow them to navigate toward their targets safely. This paper introduces a hybrid approach method for a multi-robot system, alongside with a case study in drilling and blasting automation. A Centralized Control Unit delegates tasks and information among the robots in the system, each equipped with a decentralized motion planner that supports cooperative inter-robot collision avoidance. The proposed system inherits the advantage of a centralized multi-robot system in providing a time-wise optimal solution; while also possessing the computational benefit and scalability of a decentralized system. Simulations were conducted to validate the proposed method and discuss insights into the efficacy and performance of the proposed method.**

I. INTRODUCTION

Recently, there has been an increase in attention towards applying robotics in the mining industry, with its perceived benefits to provide cost-effective solutions in environmental monitoring, efficient methods to gather information at mine sites, and improving safety for workers. These have resulted in implementations ranging from robots for exploration to human assistance, such as “Julius” - a mine site assistant robot for data collection, or “Alexander” - a mining tunnel exploration robot.

Most robotic applications are designed as a single robot system which suffers from a single point of failure. Multi-robot systems, on the other hand, are more robust and reliable against such environmental changes, presenting a potential approach for future robotic applications in mine sites. Furthermore, multi-robot systems can minimize human presence in mine sites, by promoting inter-robot coordination or human-robot interaction, with the human operating remotely.

This paper presents a hybrid design approach for a homogeneous multi-robot system, with a Centralized Control Unit (CCU) to both allocate tasks to robot agents and monitor their behavior. The individual robots, through the decentralized planners, are able to coordinate and interact with each other without explicit inter-robot communication. The proposed design inherits the advantages from both a centralized and decentralized multi-robot system, in delivering a system that has the ability to produce a time-wise optimal solution, while also processes the scalability and computational benefits for all components of the system. This system can be utilized for for different robotics applications, and for this paper, a drilling and blasting application in mining industry is used as an example case study.

All authors are with the Center for Autonomous Systems (CAS), Faculty of Engineering and Information Technology, University of Technology Sydney (UTS), Sydney, Australia. Corresponding author email: {khoa.nguyen@uts.edu.au}

Fig. 1: Example of a drilling and blasting operation with multiple machines.

Section II reviews other works related to multi-robot systems, including centralized and decentralized system design, task allocation, and motion planning. Section III provides an overview of the proposed hybrid multi-robot system, while Section IV elaborates on the simulation setup to validate the system via a case study on drilling and blasting operations performed by a team of drilling machines. Section V outlines the results from the simulations while Section VI discusses some insights into the proposed system and its efficacy. Section VII concludes the paper with brief discussion into future work to physically validate the proposed system.

II. REVIEW OF RELATED WORKS

A. Multi-Robot Systems in Mining Industry

Multi-robot Systems is a field of research that started in the early 1980s [1], motivated by the desire to improve the efficiency of existing robotic systems and achieve complex tasks that cannot by accomplished by a single robot. They are robust to changes in the environment due to their redundancy in population [1]. A multi-robot system can consist of either multiple manipulators [2], or multiple mobile robots [3], [4], or both. The mining industry has welcomed the integration of multi-robot systems to improve the efficiency and safety of existing processes [5], [6].

Typical multi-robot system architecture design can be categorized into two main approaches: centralized and decentralized systems [7]. A centralized multi-robot system consists of a Centralized Control Unit (CCU), which governs the behavior of the whole system and dictates the interaction between the robot agents [8]. Decentralized approaches focus on the behavior of the individual robots, which are based on the robot’s observation of the environment and its state [9], [10]. Hybrid approaches exist, which combine both centralized and decentralized properties into a unique architecture; however limited efforts have been conducted in this area.

B. Multi-Robot Task Allocation

Multi-robot task allocation (MRTA) involves decomposing and distributing objectives to a set of robots to achieve an overall system goal [11]. The assigned responsibility is then carried out by the robot team either through a coordinated or isolated approach. MRTA can be categorized into two subclasses based on the type of approach taken: market-based and optimization-based.

Market-based approaches are defined as mechanisms that use the robot's utilities based on a given set of tasks and constraints, and produce a solution that maximizes the overall system utility to satisfy a global team objective [11]. Most market-based task allocation systems utilize the properties of an auction in their algorithms [11], with a majority using a decentralized model in their algorithm [12]. A variant model where robots take turns in playing the role of both the producer (demanding the tasks) and the consumer (executing the tasks) was proposed by [13].

Optimization-based approaches solve the problem of task assignment using algorithms that search for an optimal solution while satisfying given objectives. Mixed integer linear programming [14], [15] allocates tasks to a team of robots to solve a particular objective. Another popular approach is genetic algorithms that have the capability to generate robust and high-quality solutions to optimization problems. The approach was applied in [16] to solve a task allocation problem using robot utility values, and [17] for the multi-robot coalition problem.

C. Multi-Robot Motion Planning

Mobile robot motion planning finds a collision-free path for the robot to move from its initial position to a target location. Motion planning can be categorized into offline and online approaches. Offline planning methods are generally applied in scenarios where the environment is known and static. Two popular examples are Probabilistic Roadmaps (PRM) [18] and Rapidly-Exploring Random Trees (RRT) [19], [20]. Online planning approaches are prevalent in systems traversing environments with limited information or dynamic environments. Notable works include Vector-field Histograms (VFH) [21] and Dynamic Window Approach (DWA) [22], which are used when an environment is observed partially. If there are moving objects, a variation of RRT for dynamic environments [23], and Velocity Obstacles (VO) [24] can be used for motion planning.

Multi-robot motion planning is similar to single robot motion planning but needs to ensure an inter-robot collision-free path for each robot in the system. Unlike single robot motion planning, multi-robot motion planning is split into coupled and decoupled planning. Coupled motion planners take into account the motions of all robots to resolve potential collisions [25], [26]. It also guarantees an optimal path for all agents, using offline motion planners. However, they are impractical for systems with a large robot population.

Decoupled motion planners, on the other hand, plan the path for a single robot using online motion planners. Unlike coupled planners, decoupled planners do not guarantee an optimal path for the robot. However, they are more robust to environmental and population size changes. Even though DWA or VO motion planning for a single robot can be applied for multi-robot systems as well, they lack the element of coordination between robots, which may result in scenarios where the robots fail to agree on certain actions for collision avoidance. Subsequently, Reciprocal Velocity Obstacle (RVO) [27] introduces coordination between robots, with each robot assuming shared responsibility in avoiding inter-robot collisions.

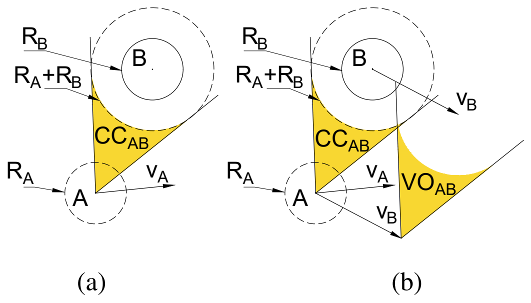


Fig. 2: RVO's derivative motion planning methods: (a) Collision Cone models potential collisions between robot, A and a static object, B by formulating a set of velocities (the colored section) that result in a collision; (b) Velocity Obstacle builds on Collision Cone to avoid collisions with a moving object, B by translating the cone using B's velocity vector.

III. METHODOLOGY

A. Proposed System Overview

The proposed multi-robot system leverages features of both the centralized and decentralized approach in designing a hybrid system. A CCU governs the behaviors of the robots, while also handling the allocation and distribution of information and data between them, while the individual robots use a local motion planner that supports inter-robot collision avoidance.

The CCU assigns tasks by evaluating the estimated time required for each robot to travel to the target location and execute the task. Once a target is received, each robot plans it own motion to move toward their assigned target while avoiding the other robots. During this process, the robots communicate with the centralized unit by sending their current state and task. While in motion, a decentralized algorithm establishes a collision perimeter around each robot. If a robot enters the collision perimeter (as communicated by the CCU), alternative motions are planned online.

Due to the large computational demand associated with creating a motion plan for large populations online in a centralized manner, a decentralized approach is utilized to handle online multi-robot motion planning. This distributes the computational load equally among the robots, reducing computational stress on the CCU and thus, supports a scalable robot population.

B. Centralized Control Unit

The Centralized Control Unit utilizes a centralized market-based task allocation system, which assigns tasks based on the individual robot's utility value. This ensures that the global objective of time to complete all tasks is optimized, whilst also allowing the tasks to be allocated dynamically to the group of robots. If any robot or group of robots finish their respective tasks, they can request new tasks to be assigned from the CCU.

The CCU collects local information from each robot such as its current state, current task assigned, and current task execution state. This information is redistributed to other robots in the system for inter-robot collision avoidance.

Unlike some other market-based approaches, the proposed CCU uses the Hungarian Algorithm [28] to assign tasks. The algorithm optimizes task allocation by minimizing the total task cost, with the CCU allocating tasks to the robots based on their utility vectors. Furthermore, since the CCU has all relevant information from the robots, the Hungarian algorithm only has to run once using the utility matrix U, decreasing computations for the CCU.

```
Result: $pose_{robot}$ == $p_{task}$
Initialization();
while $pose_{robot}$ != $p_{task}$ do
    CalcVelocityToTarget();
    states = empty();
    rvos = empty();
    if IsLowPriority() then
        for i=1:n-1 other robots do
            if InCollisionPerimeter($pose_i$) then
                $state_i$ = QueryRobotState(i);
                AddToStates($state_i$);
                $rvos_i$ = CalcRVO($state_i$);
                AddToRVOs($rvos_i$);
            end
        end
    end
    if rvos != empty() then
        CalculateNewVelocity(rvos);
    end
    SendRobotStateCCU();
end
```

Algorithm 1: The decentralized local motion planner calculating a velocity to avoid inter-robot collisions.

C. Matrix and Vector Formulation

1) Task Matrix: The task matrix is held by the CCU and is constructed from the properties and description of the given tasks. A task is defined by a location and various operation parameters. These operation parameters are defined by the CCU and are based on the chosen task, which vary in length and complexity. The task contents are analytically separated and described using numbers, which are placed in a particular order that is agreed upon by both the robots and the CCU. For a target pose, P_i and operation parameters, O_i, the task matrix for task i is defined as:

$$T = \begin{bmatrix} P_1 & O_1 \\ \vdots & \vdots \\ P_n & O_n \end{bmatrix}. \tag{1}$$

2) Utility Matrix: When the task matrix is received by the robots, each robot estimates how well it will perform each of the given tasks using local information. A utility value is determined using the estimated time to reach the target location and time to execute the assigned task. Other objectives such as the total energy consumption, or the ability of a robot to perform that task can also be used. The utility value can be used to prevent impossible allocations by setting the value to ∞, such as when a task is beyond the workspace of a robot, or when a robot is not equipped with the appropriate tool to perform a particular task.

As long as the operation parameters of the task matches the expected format, the robot will derive a utility vector $U_i = [u_1, u_2, \ldots, u_n]$ for n number of tasks. The utility vectors of m robots are collected by the CCU to form the utility matrix as follows:

$$U = \begin{bmatrix} U_1 \\ U_2 \\ \vdots \\ U_m \end{bmatrix} = \begin{bmatrix} u_{11} & u_{12} & .... & u_{1n} \\ u_{21} & u_{22} & .... & u_{2n} \\ \vdots & \ddots & \ddots & \vdots \\ u_{m1} & u_{m2} & .... & u_{mn}. \end{bmatrix} \tag{2}$$

3) Assignment Vector: The utility matrix is then used by the Hungarian algorithm [28] to produce an assignment vector A with n elements. Each element a_i represents the assignment of task i:

$$A = [a_1, a_2, \ldots, a_n], \tag{3}$$

where the *value of* a_i is the robot ID which is assigned the task, e.g. $a_2 = 5$ indicates assigning task 2 to robot 5.

D. Decentralized Robot Motion Planner

The proposed decentralized local planner utilizes the advantages of a distributed multi-robot system and eliminates the responsibility of the CCU as a motion planning unit. This approach increases the robustness of the system against any environmental or robot population change, while reducing computational demand for the CCU.

The local motion planner determines the motion for the robots to move toward the target poses, while ensuring inter-robot collision avoidance through the integration of RVO. For each robot, a user-defined collision perimeter is established that triggers the execution of the collision avoidance algorithm if another robot crosses it. The local planner also introduces a priority system for all robots within a collision perimeter, which is achieved using a weighted version of RVO. A robot is considered to be at the highest priority level if:

- It has the shortest distance to the target pose; and
- That distance is shorter than the distance to the closest robot within the collision perimeter.

Other robots within the collision perimeter are considered to be at the same lower priority. This allows the prioritized robot to reach its target without performing any collision avoidance, since the other robots would take all responsibility in avoiding that robot. The RVO for the priority robot becomes the VO in this scenario, since it is treated as a moving object, not a robot. An overview of the local motion planner algorithm is detailed in Algorithm 1.

E. Reciprocal Velocity Obstacle

Reciprocal Velocity Obstacle (RVO) is a decoupled, local reactive collision avoidance algorithm that supports coordination between robots by embedding shared responsibility to avoid each other. RVO was developed from Velocity Obstacle (VO), which models a potential collision between a robot and a moving object. A drawback of VO is that it does not take into account the autonomy of the obstacle, such as two robots using identical decentralized motion planners in the same multi-robot system. RVO, instead, solves this particular problem of VO by introducing a responsibility sharing system which assumes that each robot takes some responsibility in avoiding each other.

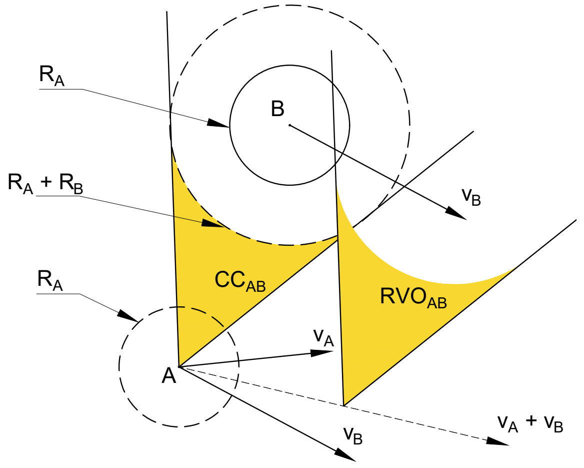
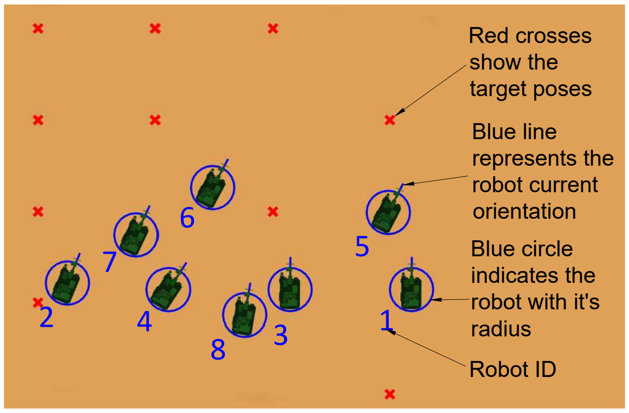


Fig. 3: Reciprocal Velocity Obstacle translates the collision cone by the average velocity vector between A and B. This process is conducted simultaneously by both robots, assuming a shared responsibility to avoid each other.

RVO has a property that can be used to enhance the level of coordination between each robot member in the system. The position of an RVO cone is defined as $0.5(v_A + v_B)$ or $(1 - \alpha_B^A)v_A + \alpha_B^A v_B$, with $\alpha_B^A = 0.5$ as the weighted parameter and v_A, v_B representing the current velocity of A and B, respectively. Consequently, altering α_B^A will shift the RVO by the sum of the weighted velocities of both A and B. With $\alpha_B^A = 1$, the position of the RVO becomes $0 + v_B$, which is equivalent to the VO of A and B. VO only assumes the responsibility of avoiding inter-robot collision



Red crosses show the target poses
Blue line represents the robot current orientation
Blue circle indicates the robot with it's radius
Robot ID
6
7
5
2
4
3
1
8

Fig. 4: An initialization of a simulation run with a group robots perform drilling at a set of randomly generated targets.

held by A, meaning that it takes B no effort to avoid inter-robot collision with A. This property is utilized to create the mentioned weighted version of RVO, which is integrated into the decentralized motion planner to leverage a higher level of coordination in the proposed multi-robot system.

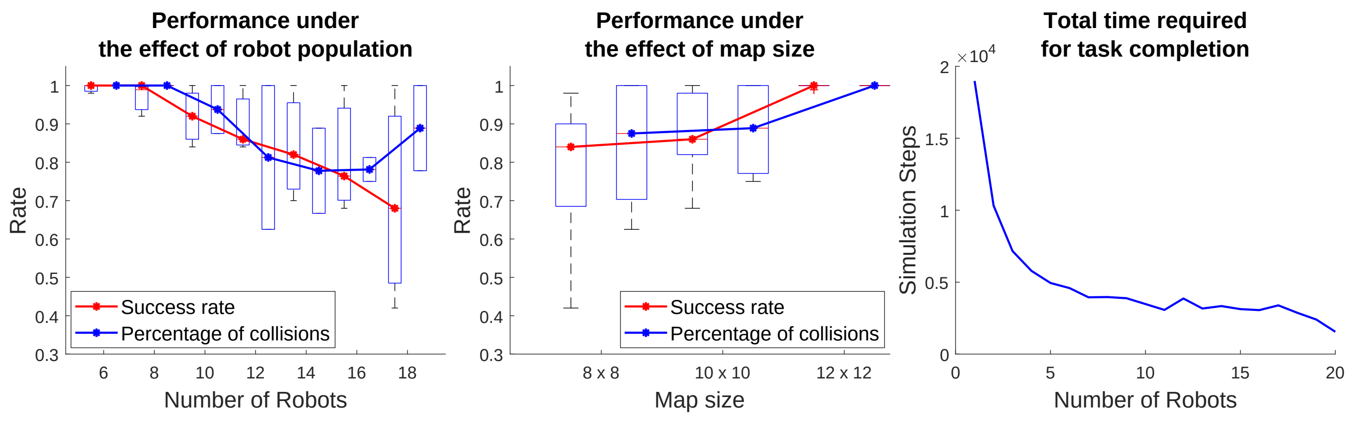
IV. SIMULATION SETUP

The simulation of the proposed system is set up in Mathworks MATLAB, using the internal Mobile Robot Simulation Toolbox and Peter Corke's Robotic Toolbox [29] for visualization and simulation purposes. The simulations present a mining case study focusing on drilling and blasting operations, where a known number of drilling points is given initially at each target pose.

For the simulations, the operation parameters for the task matrix include the number of holes to drill w, the drilling positions $D = [x_1, y_1, \ldots, x_m, y_m]$, and depth of the holes $Z = [z_1, \ldots, z_m]$. Any elements in the operation parameters which are not filled due to variations are set to 0. The simulation is setup in an open space scenario where the task target poses and the initial robot poses are initialized randomly. The minimum distance between each pose is constrained to be $\geq 4\times$ radius of the robot (R_i) to prevent an unsolvable initialization, such as when robots are randomly spawned to be in a mutually exclusive race condition.

A simulation run is treated as a failed run when, given k number of simulation time steps, one of the following occurs: (a) the multi-robot system fails to complete all tasks, including reaching the targets and executing drilling; or (b) any robot collides with another robot. For all tests, the number of simulation steps is set as $k = 20000$. The simulations are conducted to explore the following factors:

- Given a fixed map size, how does the robot population present affect the performance of the overall system (in terms of success rate and total time taken);
- How does the map size affect the performance of the overall system (in terms of success rate); and
- Given a failed simulation run, what is the most likely cause of failure (whether through an unsolvable scenario or through inter-robot collision).



(a) (b) (c)

Fig. 5: The results of the simulations showing the overall success rate and the percentage of failed simulation runs caused by robot collisions for: (a) Scenario 1 under the effect of map size; (b) Scenario 2 under the effect of robot population; and (c) the total simulation steps k taken to complete the simulations for Scenario 3.

Three main scenarios were simulated. In all simulations, the number of drilling points for each target $m = 3$, the radius of the robot $R_i = 0.25m$, and the distance between initialized poses were set to $5 \times R_i$. The numbers used within the context of this paper are for simulation purposes only. In a real world scenario, the values such as the radius of the robots or map size can be scaled up or changed accordingly.

In the first scenario, an equal number of robots and tasks are initialized at random in a fixed map size of 8m×8m. The simulations gradually increase the number of robots r and tasks n from 6 $\rightarrow$ 18. In total, 700 simulation runs were conducted with 100 runs for each set of robots and tasks.

In the second scenario, the size of the map is varied while the robots and tasks are initialized using the same parameters as in the first scenario. A total of 1050 runs were conducted with 350 runs for each map size.

In the third scenario, the map size is fixed at 10m×10m with a fixed number of tasks $n = 20$. The number of robots is varied from 1 $\rightarrow$ 20. For this scenario, 300 runs were conducted with 15 runs for each number of robots.

V. RESULTS

From Figure 5(a), it is evident that the overall performance of the system decreases when the number of robot increases, reaching its lowest success rate of approximately 0.65 with 18 robots. Moreover, the number of inter-robot collision failure cases also reduces in relation to the size of the robot population. From 5(a) 100% of all failure cases at 6 robots are due to inter-robot collisions, which drops to approximately 77% at 16 robots. In contrast, Figure 5(b) shows an increase in the performance of the system, based on the size of the map. The mean success rate rises from 85% at a map size of 8×8, and approaches 100% at the map size of 12×12. Only one failure case is observed with the largest map size, caused by an inter-robot collision. Figure 5(c) shows an overview of the time taken to complete the tasks as the number of robots increases. There is a sharp decrease in the simulation time steps taken when the number of robots rises from 1 to 6 robots. Then, the temporal improvements plateau between 6 to 17 robots, before dropping slightly as the number of robots approach the number of tasks.

VI. DISCUSSION

The results obtained from the simulations of the case study scenario provide several insights about the overall performance of the system.

Firstly, it is evident that the number of robots and the size of the map affects the efficiency of the system. As the map expands, the overall success rate increases accordingly, while a larger robot population impedes the successful execution of the tasks. This is a reasonable outcome since the chances of inter-robot collisions will increase for the same map size.

Secondly, inter-robot collisions are evidently the major reason for unsuccessful simulation runs. However, as more robots are put into a common area, the likelihood that the robots would run into an unsolvable case increases noticeably (given that the number of tasks is equivalent to the number of robots), from approximately 0% at 6 to 8 robots to nearly 23% of all failed simulations at 14 to 16 robots.This explains why the percentage of failed runs caused by collision drop in relation to the size of the robot population.

Finally, given the plateau in temporal improvements as the number of robots increase, it is unlikely that a larger robot population would result in optimal execution time. While such value eventually reaches its minimum as the number of robots approaches the number of tasks, the differences between two adjacent time instances within the interval of 7 to 19 robots are not significant, compared to those within 1 to 6. Furthermore, from the aforementioned discussion, the performance decreases in relation to the size of the robot population. Therefore, a small to medium size multi-robot system may outperform a large multi-robot system in known map size (with the system size determined by the ratio between the number of robots and the number of tasks). As a result, it depends upon how the designers of a multi-robot system weight their priorities of time, system efficiency and budgetary constraints to decide the optimal number of robots based on the given task information and their resources.

VII. CONCLUSIONS

In conclusion, this paper introduces a hybrid approach to designing a multi-robot system for drilling and blasting automation. The system is a combination of a Centralized Control Unit, which monitors the behaviors of the system's robots and delegates tasks among them, with a decentralized motion planner that ensures cooperative inter-collision avoidance between the robots. Simulations were conducted to validate the performance of the system under different scenarios. The insights drawn from the collected simulation data are summarized into three main points. Firstly, the major cause of failure is inter-robot collision, while the chance of occurring an unsolvable case increases slight with the expansion of the robot population size. Secondly, a small-medium sized multi-robot system may do as well, if not outperform, a larger multi-robot system based upon the map size. Finally, given a specific task, the designers of a multi-robot system need to balance their priorities of time, system efficiency, and budgetary constraints, to conclude the optimal number of robots in the designed multi-robot system.

ACKNOWLEDGMENT

This research is conducted in collaboration with Ausdrill from the Perenti group, and was instigated in part by DXC. We also thank UTS:RI and UTS:CAS for providing the resources required to carry out this research.

REFERENCES

[1] Y. Cai and S. X. Yang, "A Survey on multi-robot systems," in *World Automation Congress 2012*, 2012, pp. 1–6.

[2] M. Hassan, D. Liu, and G. Paul, "Collaboration of Multiple Autonomous Industrial Robots through Optimal Base Placements," *Journal of Intelligent and Robotic Systems: Theory and Applications*, vol. 90, no. 1-2, pp. 113–132, 5 2018.

[3] G. Paul and D. K. Liu, "Replanning of multiple autonomous vehicles in material handling," in *2006 IEEE Conference on Robotics, Automation and Mechatronics*, 2006.

[4] D. Liu, X. Wu, G. Paul, and G. Dissanayake, "Case studies on an approach to multiple autonomous vehicle motion coordination," *Journal of Wuhan University of Technology*, vol. 28, no. 1, pp. 26–31, 2006.

[5] R. G. Simmons, D. Apfelbaum, W. Burgard, D. Fox, M. Moors, S. Thrun, and H. L. S. Younes, "Coordination for Multi-Robot Exploration and Mapping," in *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*. AAAI Press, 2000, pp. 852–858.

[6] I. D. Miller, F. Cladera, A. Cowley, S. S. Shivakumar, E. S. Lee, L. Jarin-Lipschitz, A. Bhat, N. Rodrigues, A. Zhou, A. Cohen, A. Kulkarni, J. Laney, C. J. Taylor, and V. Kumar, "Mine Tunnel Exploration using Multiple Quadrupedal Robots," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2840–2847, 9 2019.

[7] A. Gautam and S. Mohan, "A review of research in multi-robot systems," in *2012 IEEE 7th International Conference on Industrial and Information Systems, ICIIS 2012*, 2012.

[8] B. Khoshnevis and G. Bekey, "Centralized sensing and control of multiple mobile robots," *Computers and Industrial Engineering*, vol. 35, no. 3-4, pp. 503–506, 12 1998.

[9] L. E. Parker, "ALLIANCE: An architecture for fault tolerant multi-robot cooperation," *IEEE Transactions on Robotics and Automation*, vol. 14, no. 2, pp. 220–240, 1998.

[10] T. Fukuda, T. Ueyama, Y. Kawauchi, and F. Arai, "Concept of cellular robotic system (CEBOT) and basic strategies for its realization," *Computers and Electrical Engineering*, vol. 18, no. 1, pp. 11–39, 1 1992.

[11] M. Bernardine Dias, R. Zlot, N. Kalra, and A. Stentz, "Market-based multirobot coordination: A survey and analysis," *Proceedings of the IEEE*, vol. 94, no. 7, pp. 1257–1270, 2006.

[12] B. P. Gerkey and M. J. Matarić, "Sold!: Auction methods for multi-robot coordination," *IEEE Transactions on Robotics and Automation*, vol. 18, no. 5, pp. 758–768, 10 2002.

[13] A. Chavez, A. Moukas, and P. Maes, "Challenger: A Multi-Agent System for Distributed Resource Allocation," in *Proceedings of the first international conference on Autonomous agents - AGENTS '97*. New York, New York, USA: ACM Press, 1997, pp. 323–331.

[14] N. Atay and B. Bayazit, "Mixed-Integer Linear Programming Solution to Multi-Robot Task Allocation Problem," *All Computer Science and Engineering Research*, 1 2006.

[15] M. A. Darrah, W. M. Niland, and B. M. Stolarik, "Multiple UAV dynamic task allocation using mixed integer linear programming in a SEAD mission," in *Collection of Technical Papers - InfoTech at Aerospace: Advancing Contemporary Aerospace Technologies and Their Integration*, vol. 4, 2005, pp. 2324–2334.

[16] J. Chen, Y. Yang, and Y. Wu, "Multi-robot task allocation based on robotic utility value and genetic algorithm," in *Proceedings - 2009 IEEE International Conference on Intelligent Computing and Intelligent Systems, ICIS 2009*, vol. 2, 2009, pp. 256–260.

[17] A. Rauniyar and P. K. Muhuri, "Multi-robot coalition formation problem: Task allocation with adaptive immigrants based genetic algorithms," in *2016 IEEE International Conference on Systems, Man, and Cybernetics, SMC 2016 - Conference Proceedings*. Institute of Electrical and Electronics Engineers Inc., 2 2017, pp. 137–142.

[18] L. E. Kavraki, P. Švestka, J. C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[19] S. M. Lavalle and S. M. Lavalle, "Rapidly-Exploring Random Trees: A New Tool for Path Planning," 1998.

[20] M. Clifton, G. Paul, N. Kwok, D. Liu, and D. Wang, "Evaluating performance of multiple rrts," in *2008 IEEE/ASME International Conference on Mechtronic and Embedded Systems and Applications*, 2008, pp. 564–569.

[21] J. Borenstein and Y. Koren, "The Vector Field Histogram—Fast Obstacle Avoidance for Mobile Robots," *IEEE Transactions on Robotics and Automation*, vol. 7, no. 3, pp. 278–288, 1991.

[22] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robotics and Automation Magazine*, vol. 4, no. 1, pp. 23–33, 3 1997.

[23] D. Connell and H. Manh La, "Extended rapidly exploring random tree–based dynamic path planning and replanning for mobile robots," *International Journal of Advanced Robotic Systems*, vol. 15, no. 3, p. 172988141877387, 5 2018.

[24] A. Chakravarthy and D. Ghose, "Obstacle avoidance in a dynamic environment: A collision cone approach," *IEEE Transactions on Systems, Man, and Cybernetics Part A:Systems and Humans.*, vol. 28, no. 5, pp. 562–574, 1998.

[25] G. Sharon, R. Stern, A. Felner, and N. R. Sturtevant, "Conflict-based search for optimal multi-agent pathfinding," *Artificial Intelligence*, vol. 219, pp. 40–66, 2 2015.

[26] G. Wagner and H. Choset, "M*: A complete multirobot path planning algorithm with performance bounds." Institute of Electrical and Electronics Engineers (IEEE), 12 2011, pp. 3260–3267.

[27] J. van den Berg, M. Lin, and D. Manocha, "Reciprocal Velocity Obstacles for Real-Time Multi-agent Navigation," in *ICRA*, 2008, pp. 1928–1935.

[28] H. W. Kuhn, "The Hungarian method for the assignment problem," in *50 Years of Integer Programming 1958-2008: From the Early Years to the State-of-the-Art*. Springer Berlin Heidelberg, 2010, pp. 29–47.

[29] P. Corke, "MATLAB toolboxes: Robotics and vision for students and teachers," *IEEE Robotics and Automation Magazine*, vol. 14, no. 4, pp. 16–17, 12 2007.