Towards Autonomous Weapons Movement on an Aircraft Carrier: Autonomous Swarm Parking

James Hing(✉), Kyle Hart, and Ari Goodman

Naval Air Warfare Center – Aircraft Division, Lakehurst, NJ, USA
{james.hing, kyle.m.hart, ari.b.goodman}@navy.mil

Abstract. To maintain weapons throughput on Nimitz class aircraft carriers, aviation ordnance is transported from the magazines to the flight deck and loaded onto aircraft in a process called "Strike Up". This time and labor intensive process requires multiple sailors to push weapon skids through the aircraft carrier to staging area on the flight deck.

Augmenting sailor tasking through the use of robotic equipment is one method to improve sortie generation rates (launching of aircraft), optimize manpower, and lower risk to sailors. Seen in Fig. 1, weapons skids spend extended time parked in various locations along their routes such as on elevators or in portions of hangar decks. This work presents improvements to the authors' previous work [1] to develop a Human Machine Interface (HMI) and the appropriate control methods toward supervisory control for parking multiple robotic skids in a cluttered and dynamic environment.

The HMI consists of four parts: (1) the user interface, (2) automated definition and conflict free assignment of parking goals within a user defined parking boundary, (3) collision free navigation with multiple differential drive robotic vehicles, and (4) an external infrastructure-free approach to localization and mapping.

Distribution Statement A – Approved for public release; distribution is unlimited, as submitted under NAVAIR Public Release Authorization YY-2018–28.

Keywords: Non-holonomic · Artificial potential fields · Autonomous parking

1 Introduction

High weapons throughput for Nimitz class aircraft carriers is one of many important steps to maintaining a sufficient rate of aircraft launches (sortie generation rates). To help maintain this throughput, aviation ordnance is transported from the magazines to the flight deck and loaded onto aircraft. This process, called "Strike Up", can take a significant amount of time as sailors push weapons skids through a circuitous route from the magazines to a staging area on the flight deck called the "Bomb Farm". Multiple sailors are needed to move a single heavy loaded skid, and as Fig. 1 shows, at any moment, there can be many skids on deck.

The aircraft carrier deck is not only busy with weapons movement, but it is also a dangerous environment. Sailors are continually completing tasks in close proximity to

S. Yamamoto and H. Mori (Eds.): HIMI 2018, LNCS 10905, pp. 403–418, 2018.
https://doi.org/10.1007/978-3-319-92046-7_34

Fig. 1. Weapon skids on elevator (Navy.mil Photos)

aircraft launching, recovering, and taxiing around the deck. The close proximity to aircraft puts anyone on deck at risk of bodily injury, whether it is from manual labor, loud noises, or accidents.

Optimizing and augmenting sailor tasking through the use of robotic equipment is one method of lowering the risk of bodily harm to the sailors. Specific to weapons transport, robotic transporters have been touted as a way of helping to improve the sortie generation rate and optimize manpower. An example advantage of an autonomous transport system would be for a single operator to control multiple skids all at once. In this context, we refer to multiple skids (more than three) as a swarm. In previous works on swarm control, little focus has been on the action of autonomously parking multiple systems in a particular formation. Seen in Fig. 1, weapon skids spend much time parked in various locations along their routes such as on elevators or in portions of hangar decks. This work presents improvements on previous work [1] to develop a Human Machine Interface (HMI) and the appropriate control methods to enable supervisory control for parking multiple skids in a cluttered and dynamic environment. This system could reduce the time required for a single operator or multiple operators to move the skids and setup in or exit from areas such as elevators and storage areas.

The HMI consists of four parts: (1) Automatically defining parking goal configurations for each weapon skid within a boundary, (2) the control methods for moving multiple non-holonomic weapon skids, (3) a user interface, and (4) a localization and obstacle detection method for the robotic systems.

The rest of this paper is organized as follows: Sect. 2 provides a brief overview of related work in this area. Section 3 describes the four parts of the HMI system. Section 4 presents simulation and hardware test results. Section 5 concludes the paper with a discussion and future work.

2 Related Works

Formation control of multiple robotic vehicles (swarms) is a very active area of research. Many different strategies have been developed for controlling formations of swarms under different scenarios such as movement of a formation in a corridor or

amongst obstacles. A majority of these control strategies can be categorized as either leader-follower [2], behavior-based [3], or virtual structure approach [4]. Multiple works have utilized a potential field approach to maintain the formation of a swarm of non-holonomic robots while moving to a target location [5–7]. Those works focused on maintaining a consistent formation during movement or maintaining a consistent formation on a predefined contour line. Multiple works have focused on mobile robotic platforms during parking [8, 9], but those works did not address issues of parking of multiple non-holonomic robotic platforms at the same time in close proximity to one another.

The example of weapons movement on a carrier demonstrates a Navy specific scenario where parking occurs multiple times during transport and there are multiple skids. While the works listed above present strategies for the movement of a swarm formation from point A to point B, they are not focused on the parking of formations. A parking task involves identifying safe spaces within an area enclosed by a contour. In this work, "safe" means clear of obstacles. A contour could be a physical construct such as walls or painted lines on the ground, or a virtual construct such as an operator defined virtual boundary. Ekanayake and Pathirana [10] developed a scalable control algorithm to navigate a group of mobile robots into a predefined shape and spread them inside while avoiding inter-member collisions. However, each robot was treated as an omnidirectional point mass with each robot having the same mass and mobility. Their work also did not consider dynamic obstacles within the environment.

This work builds off of previous work presented by the authors in [1]. In [1], a convex optimization approach is presented for parking multiple heterogeneous weapon skids within a user defined convex boundary. In this new work, methods for assigning parking goals are presented that enable automated parking assignment in non-convex boundaries. The previous work assigned robotic vehicles to optimized goals sequentially which can lead to robots blocking each other from getting to their assigned goals when operating in tight spaces. This new work presents a method for adaptively reassigning robotic vehicles to appropriate goals using the Hungarian algorithm with a formation driven cost function. We also present a path planning and navigation control method that overcomes some of the local minimum challenges listed in [1]. Part of the new navigation method includes a modified Reciprocal Velocity Obstacle approach that uses assigned priorities, enabling those robotic vehicles with higher priorities to move more freely through a formation. The previous interface has been revamped and all operations are now conducted on a Ubuntu computer using the Robot Operating System (ROS) toolset [11]. Notably, there is no longer a need for an overhead camera to extract positions of each robotic vehicle as localization and mapping are all performed onboard the robot using onboard sensors.

3 Autonomous Skids Parking System

Improvements on the work in [1] are implemented to address limitations. Changes to the four components of the HMI system are highlighted in the following sections.

3.1 User Interface

The two panels of the user interface are shown in Figs. 2 and 3. The goal of the user interface is to enable control of one sailor the ability to select assets in one location of the ship and to command them to move to other locations on the ship in a supervisory function. This means that the operator is not directly controlling the robots to move forward, left, right, etc. but is instead commanding end goals and then monitoring the robotic vehicles as they autonomously drive to the goals. The interface was generated using toolsets available as part of the Robot Operating System (ROS) on Ubuntu. In particular, rqt and RViz tools were used to generate the interface and visualization.

The main functionality of the interface is to give the operator situational awareness of the current pose of each of the robotic vehicles, the map of the environment, the location of detected obstacles, and awareness of the planned path for each robotic vehicle. An example of the interface can be seen in Fig. 3. The operator can select robots either by clicking on them individually on the "map view" or by dragging a selection boundary around the robots of interest. The parking boundary is then selected by the operator by drawing a selection boundary on the "map view". The operator can select if they want the goals automatically distributed using a Stacking, Convex Optimization, or Potential Field method. The operator can also individually assign goals by selecting a single robot and then clicking on the desired parking goal.

After a central controller assigns the goals, each robot's controller plans the path to the goal and commands the robot to follow that path. At any moment, an operator can cancel the paths of all robots by selecting the emergency stop "eSTOP" button or they can select each individual robot to stop. The operator can also command a new goal. Additionally, the operator can take control of any robot and drive it directly via a joystick/game pad.

3.2 Parking Goals Within Non-convex Boundaries

In [1], the authors present a convex optimization method for assigning goals for weapon skids within a user defined convex boundary. The convex boundary requirement significantly limits the operator's parking options. In the case of parking skids on an aircraft carrier, there are scenarios where parking within non-convex boundaries would be advantageous. For example it is beneficial to park around obstacles. To enable this capability, a stacking method and a potential field method is implemented for assigning goals within non-convex boundaries.

Each of the following methods use a boundary defined by the operator which is discretized into a series of position coordinates. Inputs to the method are, the number of vehicles to park, two dimensional bounding boxes of the vehicles, and desired end orientation. Currently, the desired orientation is an input rather than an output of the system for reasons described in Sect. 3.2.2.

3.2.1 Stacking Parking Method

The stacking method is a simple method of moving from the left most location of the boundary and working along the rows and columns of the boundary to place the goals. Each goal is chosen if there is no overlap between the goals and any previously placed

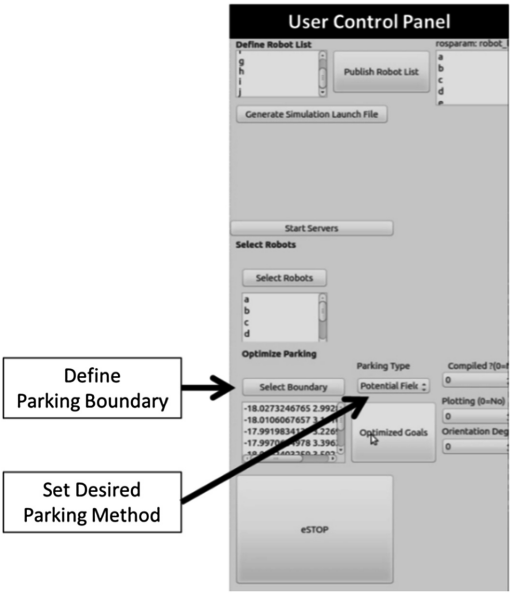

Fig. 2. User control panel of the HMI. Enables user selection of robots, selection of parking boundary, selection of parking method, and emergency stop.

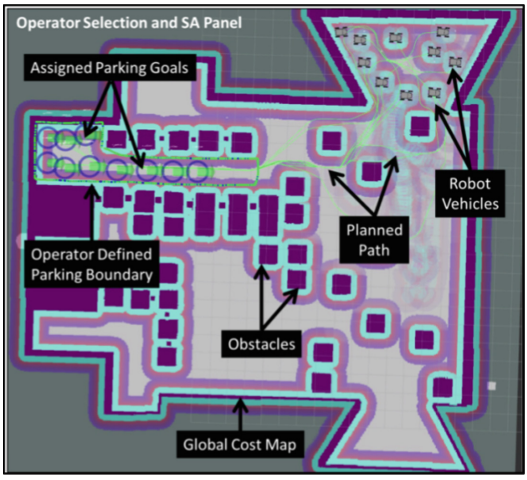

Fig. 3. Selection and situational awareness panel of the HMI. Showing example of group of robot vehicle's path plans to assigned parking goals within a user defined boundary. Environment is a mock representation of a section of a crowded hangar bay. (Color figure online)

goal (with a safety margin) such that a skid occupying the goal location would not collide with another skid. An example of the stacking method within a user defined boundary can be seen in the left side of Fig. 4.

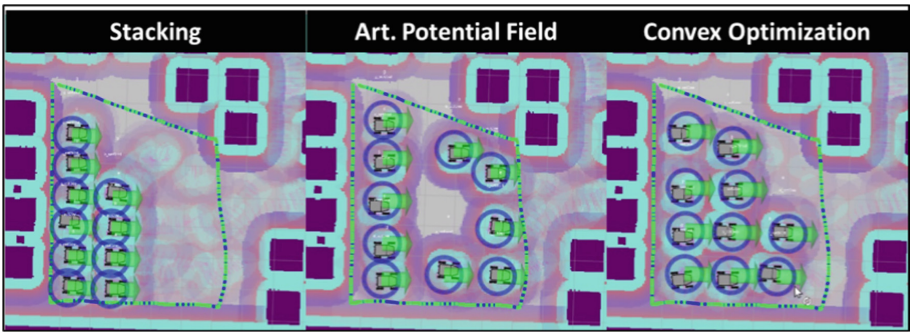

Fig. 4. Parking goal distributions. (Left) Distribution using stacking method, (Middle) Distribution using artificial potential fields, (Right) Distribution using convex optimization

The stacking parking algorithm proceeds as follows (Algorithm 1).

Algorithm 1: Stacking Method

Input: Parking boundary converted to polygon $Poly_{boundary}$, Robot geometries converted to polygon, $Poly_{robot}(i)$, and desired orientation, θ for N robots.

Output: World reference parking goals for each robot $p_{Goal}(i)$

for $i \leftarrow 1$ **to** N ***do***
for $stepX \leftarrow \min(Poly_{boundary})$ **to** $\max(Poly_{boundary})$
for $stepY \leftarrow \min(Poly_{boundary})$ **to** $\max(Poly_{boundary})$
  if $(stepX, stepY)$ inside $Poly_{boundary}$
    for $j \leftarrow 1$ **to** $(i-1)$
      if $(stepX, stepY)$ inside $Poly_{robot}(j)$
        continue to next iteration
      else
        $p_{Goal}(i) = (stepX, stepY, \theta)$
      end if
    end for
  else
    continue to next iteration
  end if
end for
end for

3.2.2 Artificial Potential Field Method

The artificial potential field (APF) method takes the stacking method a step farther. Starting from the parking results of the stacking method, the APF method further distributes the parking goals within the boundary by minimizing the total energy of the parking goals within the defined boundary as defined by

$$U_{Total}(q) = \sum U_{boundary}(q) + \sum U_{pGoal}(q) \tag{1}$$

where U_{Total} is the potential field around the centroid of the parking goal, q. U_{pGoal} is the potential function of the parking goal centroids *pGoal*. Each defined parking goal (centroid) is treated as a "floating" point. Each point on the discretized parking boundary, and other *pGoal*s act as repulsive potentials to the floating parking points whose potentials are defined by

$$U_{boundary} = \begin{cases} 0 & q > dist\ thresh \\ dist(q, boundary)^{-1} & q \leq dist\ thresh \end{cases} \tag{2}$$

$$U_{pGoal} = dist(q, p, Goal)^{-1} \tag{3}$$

where $U_{boundary}$ is the potential function of all the discretized points along the user defined parking boundary and it only has value when q is within a defined distance threshold, *dist thresh*. *dist* is the Euclidean distance function.

The induced force driving the location of each parking goal centroids is defined by

$$F_{PGoal} = \nabla U(q) \tag{4}$$

After a few iterations, the parking goals move away from the boundary points and each other. With enough time, the parking goals settle into a minimum energy point and stop moving. It is important to note that the minimum energy point could be a local minimum and not the global minimum. After settling, these points are determined to be the parking goals within the user defined boundary as seen in Fig. 4.

In practice, the APF method can loop through many iterations before all the spots settle into a local minimum. Usually close to a local minimum, the parking spots will only making small adjustments between iterations. To prevent long wait times, a set number of iterations is used as a threshold and the parking goals are set at the end of the iterations if a local minimum is not achieved. It would also be reasonable to set the threshold to be triggered when the change in overall energy between iterations is less than a set value.

We investigated the use of the vertices on the geometric shape of the vehicles to use as repulsive potentials. In this case, the sum of the forces on each vertex contributes to the resultant centroid force acting on the floating point parking goal. While this did achieve a minimum energy state, the resulting parking goals for the vehicles give the appearance of disorder as seen in Fig. 5. Instead, we chose to plan on a formation basis with constant orientation to increase operator's confidence.

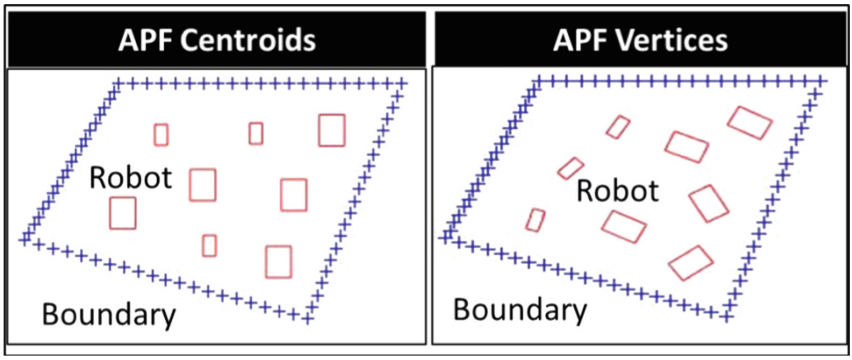

Fig. 5. Comparison between distributing parking goals using artificial potential fields via robot centroids (left) vs. robot geometric vertices (right). The robots are of varying sizes.

3.3 Adaptive Goal Assignment Using Hungarian Algorithm with Formation Driven Cost Function

In [1], final parking goals are assigned sequentially to the robot swarm, regardless of the location of those robots with respect to the parking goals. For example, parking goal One would be assigned to robot vehicle One, parking goal Two would be assigned to robot vehicle Two, and so on. This leads to inefficient movement of the robotic group within the parking boundary because no attention is paid to the amount of movement that the assigned parking goal requires from each robotic vehicle. Many times this means that there is considerable amount of avoidance maneuvers taking place to keep robots from crashing into each other within the parking boundary. Also, this leads to scenarios where one robot will reach its parking goal and end up blocking all other robots from reaching their assigned parking goals as seen in Fig. 6. To

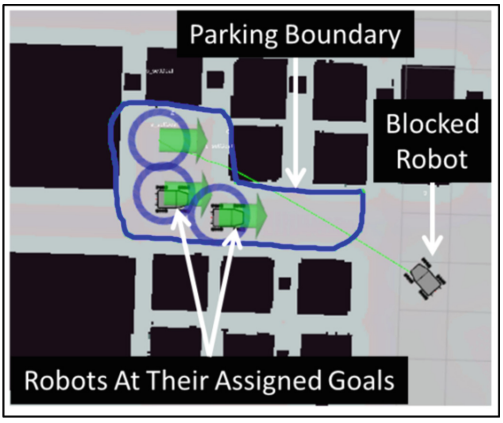

Fig. 6. Example of parked robots blocking other robots from reaching their assigned goals.

alleviate this problem, we implemented an adaptive goal assignment method that uses the Hungarian algorithm to assign the proper parking goals for the robotic vehicles.

The Hungarian algorithm is a combinatorial optimization algorithm that solves the problem of assigning m agents to n tasks [12]. It minimizes the overall costs in assignment based on a user defined cost function. In our case, the agents are the robotic vehicles and the tasks are the parking goals.

Optimization cost functions can include a precomputed estimate of distance travelled, time taken for each robot to reach the various parking goals, and/or Euclidian distance. For example, an intuitive computationally cheap approach would be to choose Euclidean distance such that the parking assignments are set based on minimizing the total distance of the group of robots from the goals. However, there are still very basic scenarios where this approach can lead to a robot blocking another robot from reaching its assigned goal, even when recalculating new assignments at every time step. A simple example of this is shown in Fig. 7.

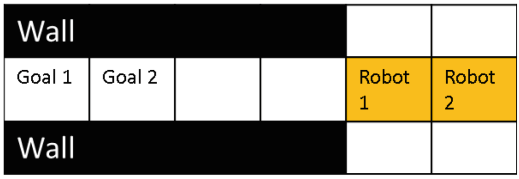

Fig. 7. Hallway example where the total cost (Euclidean Distance) for Robot 1 to Goal 2 and Robot 2 to Goal 1 is identical to the Robot 1 to Goal 1 and Robot 2 to Goal 2. It is arbitrary which goal the Hungarian Algorithm will assign with a Euclidian distance cost function, leading to cases when robots are blocked.

The cost function used in this work is based on minimizing the cost to move the group of robots from the current shape of the formation to the final shape of the parked goals at a current instant in time. It is important to note that this cost function is not dependent on the real world distances between the robots and actual parking goals. Instead, the centroid of the parking formation as a whole is superimposed on the centroid of the current robot formation. It is the Euclidean distances of the robots to these superimposed parking goals that is used in the cost function for the Hungarian algorithm goal assignments. At each instant in time, goal assignments are calculated and sent to the robotic group. This cost function alleviates the problem of robots getting to their assigned goals and blocking others from passing through.

The goal assignment algorithm proceeds as follows (Algorithm 2).

Algorithm 2: Parking Goal Assignment Method

Input: N parking goals, $p_{Goal}(i)$, N robot positions $q(i)$. A means to calculate linear sum assignment (Hungarian Algorithm) given calculated cost matrix, $cost$.

Output: $p_{Goal\_NEW}(i)$ arranged in proper order for robots, q

$offset = mean(p_{Goal}) - mean(q)$
$q_{offset} = q + offset$
for $i \leftarrow 1$ **to** N **do**
  for $j \leftarrow 1$ **to** N **do**
    $diff = p_{Goal}(j) - q_{offset}(i)$
    $cost(i, j) = norm(diff)$
  end for
end for
$goal\ indexes = linear\_sum\_assignment(cost)$
for $i \leftarrow 1$ **to** N **do**
  $p_{Goal_{NEW}}(i) = p_{Goal}(goal\ indexes(i))$
end for

3.4 Navigation with Path Planner and Reciprocal Velocity Obstacles Using Priorities

In [1], navigation is achieved through the use of an artificial potential field frame work to drive a non-holonomic robotic vehicle (differential drive system). The system works well in many cases to drive the robotic vehicles to their goals while avoiding each other. It also has the added benefit of enabling a robotic skid to drive through and out of formations. However, the downside to this method is that it only works in scenarios where a local minimum is not reached. For example, Fig. 8 shows two robots swapping positions within two formations. Potential fields will not succeed as the robots will get stuck in the corners of the rooms due to a local minimum condition.

To address this drawback, we used a path planner based on Dijkstra's algorithm [13] to plan the route for each robot to their assigned parking goals. The path planner takes in the map of the environment, the goal (i.e. assigned parking goal), and all detected obstacles, and then calculates an optimal path to the goal. Parts of robots detected by other robots are treated as detected obstacles. The potential fields approach from [1] is then used to drive the robotic vehicle to follow the path generated by the Dijkstra's algorithm as seen in Fig. 3 by the green lines extending from each robot to the assigned parking goal.

3.4.1 Reciprocal Velocity Obstacles Using Priorities

A Reciprocal Velocity Obstacle (RVO) approach using priorities is implemented to enable a robotic system to move through existing formations. An example of an RVO

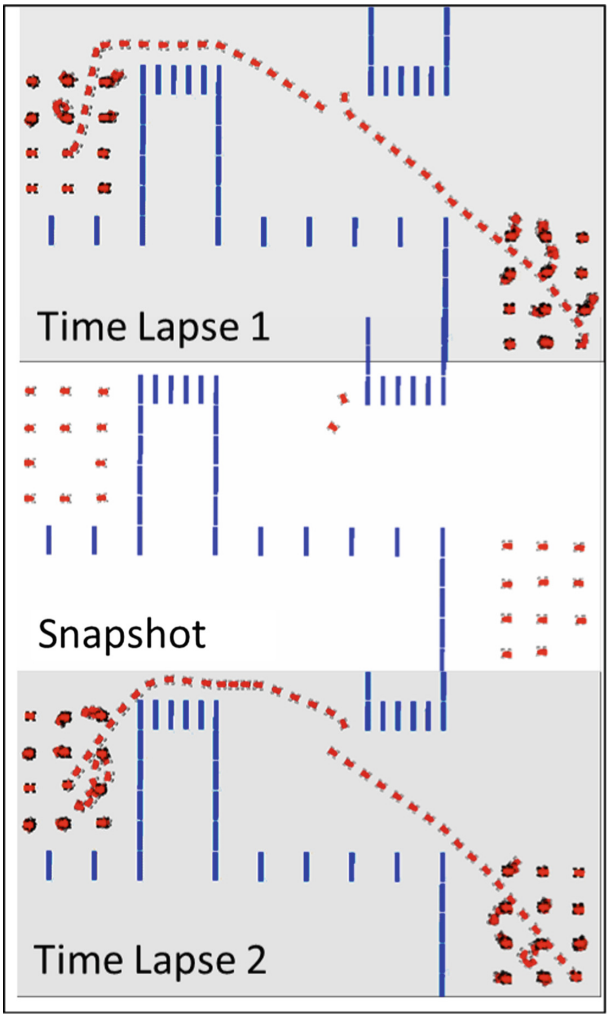

Fig. 8. Time lapse images of two robots swapping places within parking formations using RVO.

implementation is shown in Fig. 8. The reason for this capability is to handle scenarios when a single weapon skid within a formation (e.g. a skid in the middle) is needed to go elsewhere. This capability will also allow for skids or sailors to walk through the parked formation without having to move each skid individually. The skid needing to leave the formation, or needing to get through the formation, is given a high priority (i.e. more authority to occupy a space) and other skids move out of the way.

RVO is a technique that has been widely used for safe navigation among moving obstacles [14]. In very simple terms, each moving robot looks at other robot positions, velocities, and geometric shape, and uses that information to determine the best velocity to avoid a collision. The description and mathematical formulation of the RVO

algorithm can be found in [14]. In addition, they outline how the behaviors extend with priority, called general reciprocal velocity obstacles. A graphic showing a velocity obstacle determination can be seen in Fig. 9.

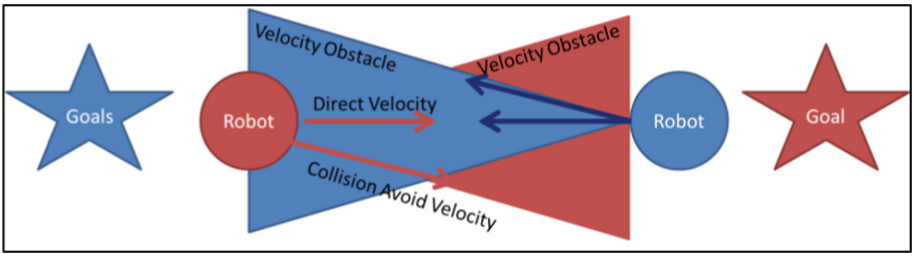

Fig. 9. Robots (circles) trying to get to their assigned goal (star). Velocity resulting from RVO is the closest velocity to the direct velocity without entering the velocity obstacles.

3.5 Robot and Obstacle Localization

In [1], positions of the robot and obstacles are extracted from images obtained from a top down camera viewing the operational environment. On an aircraft carrier, the weapon skids traverse through many areas of the ship including the hangar bay and on the deck. It would be difficult to build an external imaging infrastructure onto the ship that would allow for complete visual tracking of the robotic assets at all times. In this work, a mapping and localization approach using only onboard sensors (i.e. LiDAR, camera) to the robotic vehicle itself is implemented to eliminate the restrictions of needing a top down camera.

3.5.1 Onboard Localization and Mapping

Localization of the robotic vehicles using onboard sensors requires information about the system's operating environment. Typically, a simultaneous localization and mapping (SLAM) approach is utilized to build up information (i.e. map) about the operational environment. In this work, the SLAM approach called "GMapping" developed by [15] is used to build a map of the operational environment for the robotic systems. For weapons movement on the aircraft carrier, it is not unreasonable to believe that a rough blueprint of the various areas of the ship could be fed to the robotic skids prior to operations, thereby lessening the workload of SLAM. Along with this, the robotic skids can be continuously running GMapping to update the maps should major obstacles change (e.g. cargo pallets placed or moved).

If a general map of the operational environment exists, the robotic skids only need to localize within that map. An algorithm called Adaptive Monte Carlo Localization (AMCL) is used to localize the robotic vehicles. AMCL uses a particle filter to track the pose of the robot against the known map as described in [16].

Obstacles within the environment are detected by each robotic vehicle's onboard sensors. This obstacle information is used to update a robot's local map. A central controller uses each robot's location to manage parking assignments and sends those

assignments back to the appropriate robots. Each robot then uses its local map, which includes detected obstacles, to plan appropriate paths to the assigned goal.

3.5.2 Shared Mapping for Common Reference Frame

A common reference frame for all information being received is an important aspect of the central controller to coordinate the movement of a swarm. It will be a challenge on an aircraft carrier to guarantee the exact known location of every robot at boot up, making it difficult to define a common reference frame. When a map of the operational environment exists, the reference frame of the map is then shared by all robots that are localizing themselves within that map. However, in scenarios where a map of the environment does not exist prior to operations, a map needs to be created by the robotic systems. Using GMapping, a local reference frame to the robot creating the map is used. The central controller then needs to take this local reference frame and make it a global reference frame shared by all the robotic vehicles.

There are a few approaches that could be used to generate a global shared reference frame. A couple examples are: (1) Robotic vehicles traversing the same areas can use AMCL to localize within the map created by another robot. (2) Robotic vehicles can visually see each other from afar and can use visual features to create a commonly shared reference frame.

4 Results and Discussion

The new interface and control methods were tested in hardware using two robotic vehicles and in simulation using multiple simulated robotic vehicles. The simulation setup was demonstrated at the Department of Defense Lab day held at the Pentagon where visitors were able to select simulated robots and command them to park in user defined boundaries. The simulated environment, shown in Fig. 3 is a generalized representation of a crowded hangar bay with staging areas and elevators for the robotic skids to park on. With minimal instruction, visitors were able to use the interface with ease and were able to successfully park the simulated robots in many different types of user defined boundaries without collisions and without inter-robot obstruction. The hardware tests used two robots due to laboratory resources limitations but the operator was able to command the robotic systems around a pre-mapped environment using the developed interface without the use of any overhead camera system. An example of the hardware test is shown in Fig. 10.

RVO with priorities enabled multiple differential drive robotic vehicles to move through and into formations. An example simulation result can be seen in Fig. 8 where two robots swap locations within parked formations. It was also tested in hardware with two robots passing by each other without colliding as shown in Fig. 11. The bottom starting robot is given a higher priority than the top starting robot. The top robot takes a larger avoidance route because of this. RVO shows promise as being a good method for performing local obstacle avoidance. The RVO implementation, as written in this work, grows linearly with respect to the number of robots.

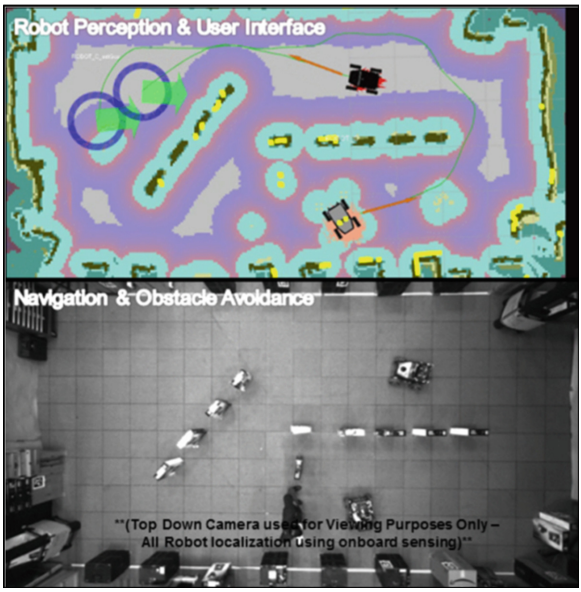

Fig. 10. Hardware demonstration of user interface and robot control. Two robots are autonomously navigating to the assigned goals.

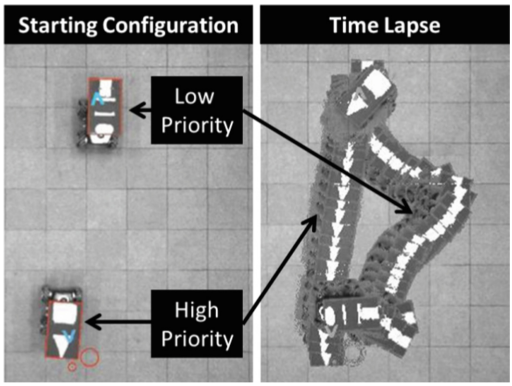

Fig. 11. Hardware demonstration of RVO. Two robots commanded to pass each other. One with higher priority that the other. (Right) Time lapse showing lower priority robot taking larger avoidance route due to RVO with priorities.

5 Conclusions and Future Work

This work presents improvements over the previous HMI and artificial potential field approach to parking multiple autonomous differential drive skids. The improvements enabled the control of multiple robotic systems by a single operator in a wider range of

environment scenarios and without the need for an overhead camera to extract pose information of the robotic systems.

Now that the majority of the functionality of the operator interface and robotic control system is in place, one of the next steps will be to conduct a formal evaluation of operator performance when controlling multiple robotic vehicles using the interface. There is also technical work needed to enable the control methods to work for operating robotic skids with different steering mechanism (e.g. Ackerman steering). The goal assignment methodology in this work requires that the geometry of all parking spots be the same size and defined by the largest bounding box of the robotic vehicles in the commanded group. This enables the adaptive shifting of parking assignments without concern that a larger robot could be assigned to a space that it can't fit into. Future work can investigate goal assignments that take into account varying robotic vehicle sizes (addressed by our APF methodology) and the appropriate sequence at which they should be filled.

In general, there are many additional elements of the autonomous vehicle system that need to be further developed before an autonomous weapon skid is fielded. As stated in [17], there are aspects of the human system interface, monitoring and diagnosis, planning and decision, sensing and perception, and networking and collaboration that have to be addressed at the vehicle management system level, the mission management system level, and the command and control system level to field a safe and successful autonomous system. This work touched on much of the vehicle management system level and a small bit in the mission management system side. There are many other elements that need to be addressed in future work to transition this technology.

References

1. Hing, J., Boczar, R., Hart, K.: Parking autonomous skids. In: Yamamoto, S. (ed.) HCI 2015. LNCS, vol. 9173, pp. 557–568. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-20618-9_55
2. Cowan, N., Shakerina, O., Vidal, R., Sastry, S.: Vision-based follow-the-leader. In: IEEE/RJS International Conference on Intelligent Robots and Systems (2003)
3. Scharf, D.P., Hadaegh, F.Y., Ploen, S.R.: A survey of space formation flying guidance and control (part 2). In: American Control Conference, Boston, MA (2004)
4. Ren, W., Beard, R.W.: A decentralized scheme for spacecraft formation flying via the virtual structure approach. In: American Control Conference, Denver (2003)
5. Lawton, J., Beard, R., Young, B.: A decentralized approach to formation maneuvers. IEEE Trans. Robot. Autom. **19**(6), 933–941 (2003)
6. Liang, Y., Lee, H.-H.: Decentralized formation control and obstacle avoidance for multiple robots with nonholonomic constraints. In: American Control Conference, Minneopolis, MN (2006)
7. Elkaim, G., Kelbley, R.: A lightweight formation control methodology for a swarm of nonholonomic vehicles. In: IEEE Aerospace Conference, Big Sky, MT (2006)
8. Kondak, K., Hommel, G.: Computation of time optimal movements for autonomous parking of non-holonomic mobile platforms. In: International Conference on Robotics and Automation, Seoul, Korea (2001)

9. Masaki, H., Kangzhi, L.: Automatic parking benchmark problem: experimental comparison of nonholonomic control methods. In: Chinese Control Conference, Hunan, China (2007)
10. Ekanayake, S., Pathirana, P.: Formations of robotic swarm: an artificial force based approach. Int. J. Adv. Rob. Syst. **7**(3), 173–190 (2010)
11. Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., Ng, A.: ROS: an open-source robot operating system. In: ICRA Workshop on Open Source Software (2009)
12. Kuhn, H.: The Hungarian method for the assignment problem. Nav. Res. Logist. Q. **2**, 83–97 (1955)
13. Dijkstra, E.W.: A note on two problems in connexion with graphs. Numer. Math. **1**, 269–271 (1959)
14. Van den Berg, J., Lin, M., Manocha, D.: Reciprocal velocity obstacles for real-time multi-agent navigation. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA) (2008)
15. Grisetti, G., Stachniss, C., Burgard, W.: Improved techniques for grid mapping with rao-blackwellized particle filters. IEEE Trans. Rob. **23**, 34–46 (2007)
16. Fox, D.: KLD-sampling: adaptive particle filters. In: Advances in Neural Information Processing Systems, vol. 14, pp. 713–720 (2001)
17. Naval Studies Board: Autonomous Vehicles in Support of Naval Operations (2005)