Parking Autonomous Skids

James Hing[1], Ross Boczar[1], Kyle Hart[1]

[1] Naval Air Warfare Center – Aircraft Division, Lakehurst, NJ, USA
{james.hing, ross.boczar, kyle.m.hart}@navy.mil

Abstract. Autonomous movement of materiel aboard an aircraft carrier can potentially be accomplished using robotic skids. This work proposes a simple human machine interface (HMI) and a control algorithm that would enable a sailor to control and park multiple robotic skids. This work specifically looks at an artificial potential field approach to parking multiple robotic skids (non-holonomic) within a user defined boundary. Optimal goal locations within the boundary for the skids are calculated through convex optimization techniques.

Keywords: Non-holonomic, artificial potential fields, autonomous parking.

1 Introduction

To support naval aviation aboard aircraft carriers, 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". The Strike-Up process is one of the major bottlenecks to increasing mission-capable sortie rate (deployment of aircraft).

Robotics weapons transporters have been touted as a way of improving sortie rate and optimizing manpower. A true advantage of an autonomous transport system would be for a single operator to control multiple systems ("swarm"). In previous works on swarm control, little focus has been on the action of autonomously parking multiple systems. Seen in Fig. 1 left, weapons skids spend much time parked in various locations along their routes such as on elevators or in portions of hangar decks. This work has developed a Human Machine Interface (HMI) and the appropriate control methods towards supervisory control for parking multiple skids in a cluttered and dynamic environment. This system will reduce the time it takes for a single operator or multiple operators to move the systems and setup in / exit from elevators or storage areas.

The HMI consists of 4 parts: 1) Automatically defining optimal parking locations for each weapon skid within a boundary, 2) the control method for moving multiple non-holonomic weapon skids, 3) a user interface and 4) extracting robot / obstacle pose.

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

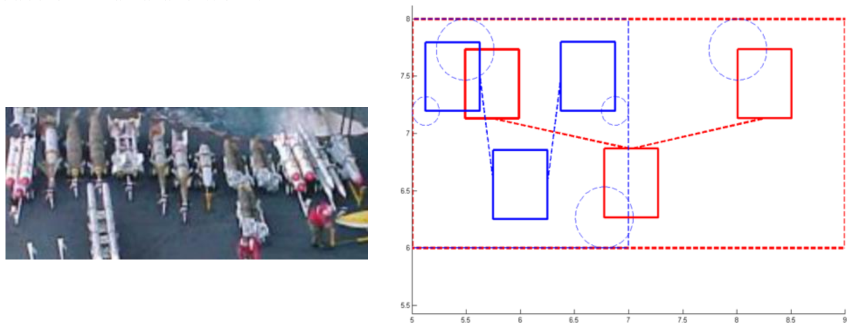

Fig. 14. Left) Example of Skids parking on an elevator, Right) Example of optimizing the spacing for three mechanized skids. (*Red/Light Grey)*: solution in large contour. (*Blue/Dark Grey)*: solution in smaller contour.

2 Related Works

Formation control of multiple robotic vehicles, also known as groups or swarms, has been a very active area of research. Robotic swarms have different advantages in different applications. In many cases, the behavior of the swarm can produce abilities that outweigh the abilities of a single member. An example would be a group of robots carrying an object of a weight heavier than the payload limit of a single robot alone. In other cases, the act of moving groups of vehicles all at once rather than individually increases performance and throughput such as in the transportation of material (which is the case study used in this work). Many different strategies have been developed for the control of these swarm formations under different kinds of 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 [1], behavior-based [2], or virtual structure approach [3].

While the works listed above present strategies for the movement of formations from point A to point B with obstacles along the way, they are not suitable for the task of parking formations of non-holonomic robotic vehicles. Less focus has been on the control of multiple non-holonomic robotic vehicles during parking tasks. Multiple works have focused on single mobile robotic platforms during parking [4] [5]. The example of weapons movement on a carrier demonstrates a Navy specific scenario where parking occurs multiple times during transport and there are multiple heterogeneous sized skids.

This work focuses on the parking aspect of swarm formation control. Specifically the focus is on a behavior-based, potential field approach for parking multiple non-holonomic vehicles in dynamic environments (i.e. moving obstacles) within a defined contour (parking space / area). This is a desirable method for real-time path planning because the computation requirements for obtaining the potential factors are low.

Multiple works have utilized a potential field approach to maintain formation of a swarm of non-holonomic robots while moving to a target location [6] [7] [8]. Those works focused on maintaining a formation (aka shape) during movement or maintaining a formation on a predefined contour line. However, for parking tasks which require optimizing the space within the area enclosed by a contour, the above methods as presented are not suitable. To our knowledge, the only work related to this aspect of optimizing the space within an area enclosed by a contour is the work by Ekanayake and Pathirana [9]. In their work, they 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. Lastly, their work did not consider dynamic obstacles within the environment.

In this work, we have greatly extended the work presented in [10] that used potential fields to maneuver a single non-holonomic rectangular robot through a static obstacle course to a goal location. While [10] only focused on a single robot and static obstacles, their practical method of local obstacle avoidance offered a good starting structure from which to extend the control and parking of multiple non-holonomic rectangular skids.

3 Autonomous Skids Parking System

3.1 Optimal Parking Locations

First we look at the problem of optimally filling a given shape with a swarm of rectangular robots. This is representative of defining the optimal parking locations of each robotic skid within an allocated parking area of the ship (e.g. elevator, hangar bay, etc.).

Formulation

Consider the problem of arranging N rectangular robots inside a given shape, with respective positions (x_i, y_i), lengths l_i, and widths w_i. We consider a swarm of planar robots in $\mathbb{R}^2$. We assume that the desired swarm formation includes consistent orientation across all robots, which we take without loss of generality to be $\theta_i = 0, \forall i$.

Floor Planning

Following the formulation set out in Chapter 8 of Boyd & Vandenberghe [11], we can pose this as floor planning problem: minimizing some objective while packing the rectangular robots into a given shape, with no overlap between robots. Using convex optimization for VLSI floor planning is common; see [12]. The objective being optimized can be the size of the shape itself or some other metric, such as intra-robot spacing. As stated, the no-overlap constraint can lead to a combinatorial optimization problem which is often computationally intractable. Thus, we need to impose some structure on the swarm configuration in order to pose a tractable convex optimization problem.

Relative positioning

Inspired by 8.8.1 of [11], the problem of reducing the combinatoric constraints can be solved by the construction of two N-node directed acyclic graphs describing the relative positioning between the robots: $\mathcal{H}$ (horizontal) and $\mathcal{V}$ (vertical). A set of tractable relative positioning constraints is predicated on the existence of a directed path between each pair of robots (represented by nodes i and j) in either $\mathcal{H}$ or $\mathcal{V}$.

Though many different schemes can be used, our method attempts to use minimal edges, though we do not currently have any proof of minimality. Two possible schemes, based on square and diamond arrangements, are seen with N = 9 in **Fig. 15**.

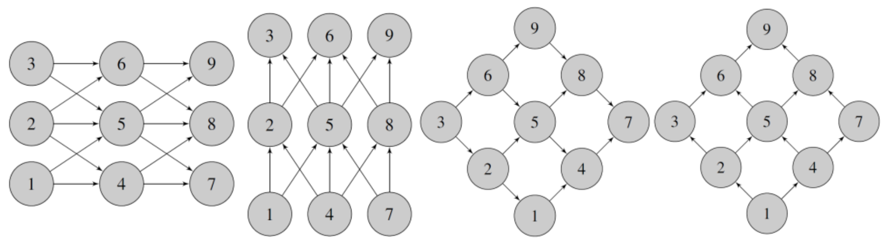

Fig. 15 1) Square horizontal relative position graph $\boldsymbol{\mathcal{H}}$, 2) Square vertical relative positioning graph $\boldsymbol{\mathcal{V}}$, 3) Diamond horizontal relative positioning graph $\boldsymbol{\mathcal{H}}$, 4) Diamond vertical relative positioning graph $\boldsymbol{\mathcal{V}}$.

These graphs describe the relative horizontal and vertical positioning between nodes: for example, node 4 is to the right of and below node 2. With a directed edge denoted by (i,j), the positioning constraints are given by:

$$x_i + l_i \leq x_j,\ \forall(i,j) \in \mathcal{H}\ . \tag{3}$$
$$y_i + w_i \leq y_j,\ \forall(i,j) \in \mathcal{V}\ . \tag{4}$$

These inequalities are linear, and are therefore convex. For our particular graph structure, the number of constraints grows as $O(N)$.

Spacing constraints

We can also impose minimum distance constraints (which could also be variables to be maximized) of the form:

$$\left\|p_i - p_j\right\|_2 \geq D_{ij}, \forall\ i \neq j, \text{i,j=1,…,N}\ . \tag{5}$$

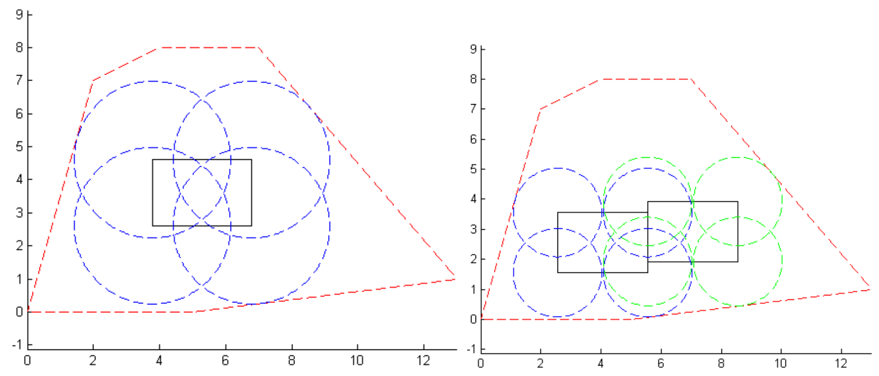

Fig. 16. Left) Optimized $\boldsymbol{L^2}$ boundary spacing. Right) Optimized swarm-minimum $\boldsymbol{L^2}$ boundary spacing.

where $p_i = \left[\frac{x_i+l_i}{2}, \frac{y_i+h_i}{2}\right]^T$, x_i, y_i is the lower left corner, and p_i is the center. Though this leads to a non-convex problem, there are several convex relaxations, restrictions, or reformulations which can be employed.
1) *Linear approximation*: As seen in [11], we can replace the constraints (5) with ones of the form:

$$a_{ij}^T(p_i - p_j) \geq D_{ij}, i < j, i, j = 1, \dots, N\ . \tag{6}$$

where $||a_{ij}||_2 = 1$. As an example, a simple heuristic for choosing a_{ij} is to first solve a feasibility program without constraints (6), with optimal solution $\widehat{p}_i$, and to then set $a_{ij} = (\hat{p}_i - \hat{p}_j)/\left\|\hat{p}_i - \hat{p}_j\right\|_2$.
2) *Relative position spacing*: We can also exploit the relative positioning constraints and change constraints (3) and (4) to:

$$x_i + l_i + \alpha_{ij} \leq x_j,\ \forall(i,j) \in \mathcal{H}\ . \tag{7}$$
$$y_i + w_i + \beta_{ij} \leq y_j,\ \forall(i,j) \in \mathcal{V}\ . \tag{8}$$

In this case, we could maximize some weighted combination of α_{ij}, β_{ij}, which represent horizontal and vertical spacing, in order to maximize some function of intra-graph spacing. Additionally, we can impose minimum distance constraints α_{min}, β_{min}.
3) *SDP relaxation*: If we consider the robots as point masses, maximizing the spacing becomes a variant of the well-known circle-packing problem. However, SDP relaxations of this problem are not known to work well in practice [13].

Bounding shape constraints and spacing maximization

For simplicity, we limit the desired shape to a convex polygon which can be expressed in the form $Az \leq b$ where $A \in \mathbb{R}^{m\times 2}$, $z \in \mathbb{R}^2$, $b \in \mathbb{R}^2$. Feasibility constraints are then of the form $Ap_{ik} \leq b$, where we denote p_{ik} as the position of the kth corner of the ith rectangular robot. These constraints can also be reduced using graphs $\mathcal{H}$ and $\mathcal{V}$. Section 8.5.1 of [11] formulates the problem of finding the largest Euclidean ball contained inside a polyhedron, given by $Az \leq b$ as:

$$\begin{aligned} &\max && R\,. \\ &\text{subject to} && a_i^T x + R\|a_i\|_2 \leq b_i, i = 1, \dots, m\,. \\ &&& R \geq 0\,. \end{aligned} \tag{9}$$

with variables x and R. Due to convexity of both the bounding shape and the rectangular robots, we can find the placement of robot j which maximizes the shortest distance between the robot and the bounding shape by solving the convex optimization problem:

$$\begin{aligned} &\max R\,. \\ &\text{subject to } a_i^T p_{jk} + R\|a_i\|_2 \leq b_i, \forall\, i, k\,. \\ &\quad R \geq 0\,. \end{aligned} \tag{10}$$

We can also impose a minimum Euclidean distance with the constraint $R \geq R_{min}$.
An example of a solution to this problem is seen in **Fig. 16** left. Thus, for a multi-robot swarm, we can maximize some concave function of these distances, which we now represent as $R = [R_1, R_2, R_j, \dots, R_N]$:

$$\begin{aligned} &\max\ \phi(R)\,. \\ &\text{subject to } a_i^T p_{jk} + R_j\|a_i\|_* \leq b_i, \forall\, i, j, k\,. \\ &\quad R \succcurlyeq 0\,. \end{aligned} \tag{11}$$

Fig. 16 right shows an example of a 2-robot swarm with no minimum spacing constraint, maximizing $\phi(R) = min_j\{R_j\}$. Assuming a constant shape, the number of constraints grows as $O(N)$.

Combined boundary/inter-robot spacing

Using both boundary spacing and intra-robot spacing constraints, we can form a convex optimization problem. For this formulation, we use relative position spacing constraints, and assume no special weighting (though this can be easily added). Thus our problem is:

$$\begin{aligned} &\max && \psi(\alpha, \beta, R)\,. \\ &\text{Subject to} && x_i + l_i + \alpha_{ij} \leq x_j,\ \forall (i,j) \in \mathcal{H}\,. \\ &&& y_i + w_i + \beta_{ij} \leq y_j,\ \forall (i,j) \in \mathcal{V}\,. \\ &&& a_i^T p_{jk} + R_j\|a_i\|_* \leq b_i, \forall\, i, j, k\,. \\ &&& \left.\begin{aligned} p_{j1} &= [x_j, y_j]^T \\ p_{j2} &= [x_j + l_j, y_j]^T \\ p_{j3} &= [x_j + l_j, y_j + w_j]^T \\ p_{j4} &= [x_j, y_j + w_j]^T \end{aligned}\right\} \forall\, j\,. \\ &&& R \succcurlyeq 0\,. \end{aligned} \tag{12}$$

The function ψ is used to quantify sums, averages, or minimums of distances. However, the objective function must remain concave, so ψ will usually be some type of “maximin” function such as (13).

$$\psi_1(\alpha, \beta, R) = \min_{(i,j)\in\mathcal{H},(i,k)\in\mathcal{V}}\{\alpha_{ij}, \beta_{ik}, R_i\}\,. \tag{13}$$

Once formalized, these optimization problems can be solved using the CVX software package [14].

Structured Optimization

Since many of these objective functions involve minimums of spacing variables, the optimization often finds an optimal solution at a “bottleneck” - a subset of robots are tightly packed across some cut of the contour, with relevant inequality spacing constraints being equality (known as *active constraints*). Thus, the placement of robots far away from the bottleneck may be closer than normal and appear suboptimal, as spacing them out further than the minimum does not improve the objective.

Thus a "structured optimization" approach may be used to iteratively improve the qualitative solution:

- $C = \emptyset, K = \{all\ constraints\}$
- $For\ i = 1 \ldots N_{iter}$:
  - Solve the optimization problem with constraints $K \setminus C$, including removing C from the objective function and holding constraints C to their fixed values.
  - Determine which constraints are active (A), and set $C = C \cup A$. Store values of constraints in C achieved at the optimal point.

3.2 Non-holonomic Skid Control

Our approach extends the artificial potential field frame work in [10] to drive multiple non-holonomic vehicles. In [10], each platform has virtual action points applied to the front and rear of the vehicle that are used to apply both attractive and repulsive forces from the goal and obstacles respectively. The attractive goal force is the tangential vector at the front action point of the robot to the radius that comes in contact with the orientation of the goal front action point. Obstacle forces behind the vehicle's pivot point produce a repulsive force at the rear action point and obstacles in front of the

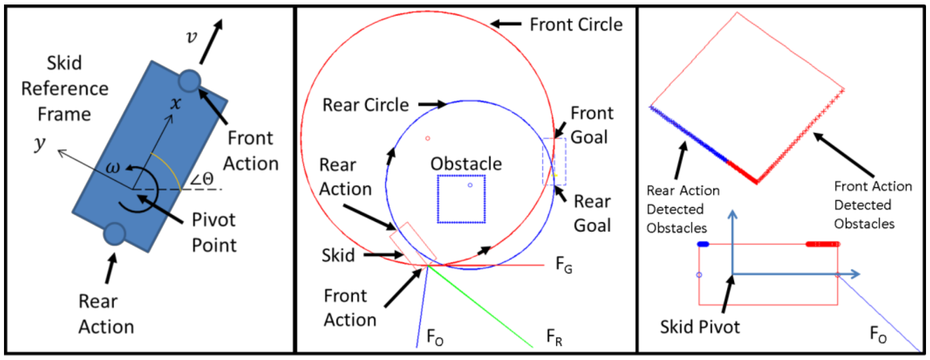

Fig. 17. Left) Reference frame for skid/robot, Middle) Top down view of the force vectors and trajectory options during travel to parked goal location. Right) Skid frame view of obstacle shown in image left.

pivot point produce a force at the front action point. All forces are proportional to the inverse of the cube distance between obstacle points and robot body. The resultant force F_R is a summation of the attractive force from the goal F_G and the forces from detected obstacles F_O. Forces acting on the rear action point are inverted and applied to the front action point during this summation. The equations for F_G and F_O can be found in [10] and are not presented here due to page limitations.

The extensions to the algorithm we have implemented are as follows:

1) [10] assumes that the front of the robot is always desired to be driven toward the goal orientation. Instead, implementing the same method for the front action point with the rear action point allows a decision to be made on which end of the vehicle to drive towards the goal. In many cases, driving the rear action point towards the goal location results in a shorter travel distance than using the front action point. This requires the calculation of the arc length for the front arc and the rear arc to determine which is the shorter path.
2) Multiple skids controlled at the same time. This is a relatively straight forward extension to [10] as each skid calculates its' own resultant force vector based on its commanded goal location, current position, and detected surrounding obstacles. Surrounding skids are simply treated as moving obstacles to each other during the force calculations however we have implemented a weighting coefficient that allows inter-skid forces to be tuned differently from other obstacle forces.
3) Addition of a weighting function to the goal force. In [10], the goal force is always given a magnitude of 1 as its importance is to define the tangential direction along the arc to the goal orientation. The addition of weighting function based on the distance to the goal has enabled the ability to move robots through formations as seen in the Results section.

The resulting command to the robot, based on the resultant force vector, is a forward or reverse speed and a rotation about the pivot point (in this particular case the pivot is the middle point of the axle between the two rear wheels):

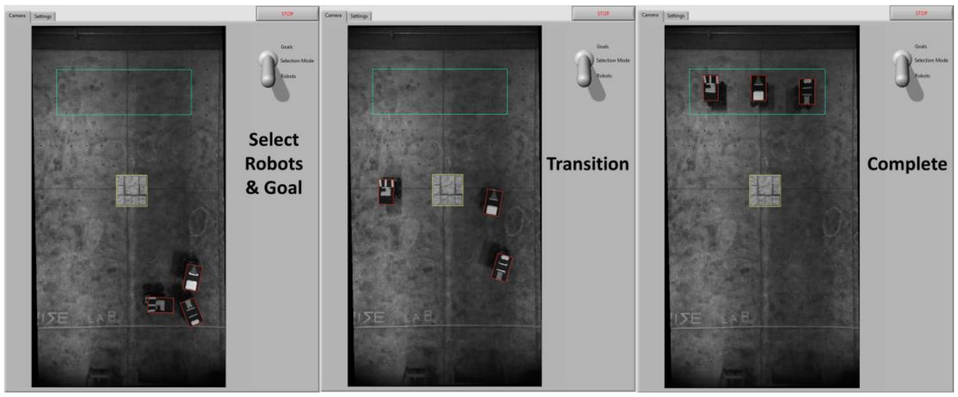

Fig. 18. Simple user interface for selecting robots, selecting the goal parking contour, and monitoring progress.

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} F_{Rx} \\ \frac{F_{Ry}}{x} \end{bmatrix}. \tag{14}$$

where v is the commanded forward or reverse speed in the skid frame, ω is the angular velocity commanded to the robot, $F_{Rx}$is the horizontal component of the resultant force vector $F_R$in the skid frame, and x is the distance (in the skid frame) to the action point (either front or rear) that the current force vector is acting on.

3.3 The User Interface

The user interface, shown in Fig. 18, is inspired by the notion of enabling a sailor to select assets (robotic skids) in one ship location and select the goal location of that asset in another location. In actual implementation it is expected that the sailor would not have an actual camera top down view but instead would have a map view of the ship. The interface allows the user to select the desired robots and define a bounding box where the assets will park. During bounding box select, the user defines the general orientation that will be applied to all systems within the boundary. Optimal spacing of the robots is automatically calculated and goal locations are sent to the control algorithm to drive the robotic systems. At any moment during travel the user can select any of the moving assets and define a new goal location. Also, if during travel, an obstacle occludes part of the parking boundary drawn by the user, a smaller boundary that does not encompass the obstacle can be defined and new parking locations defined using the same optimization algorithm. An example is shown in

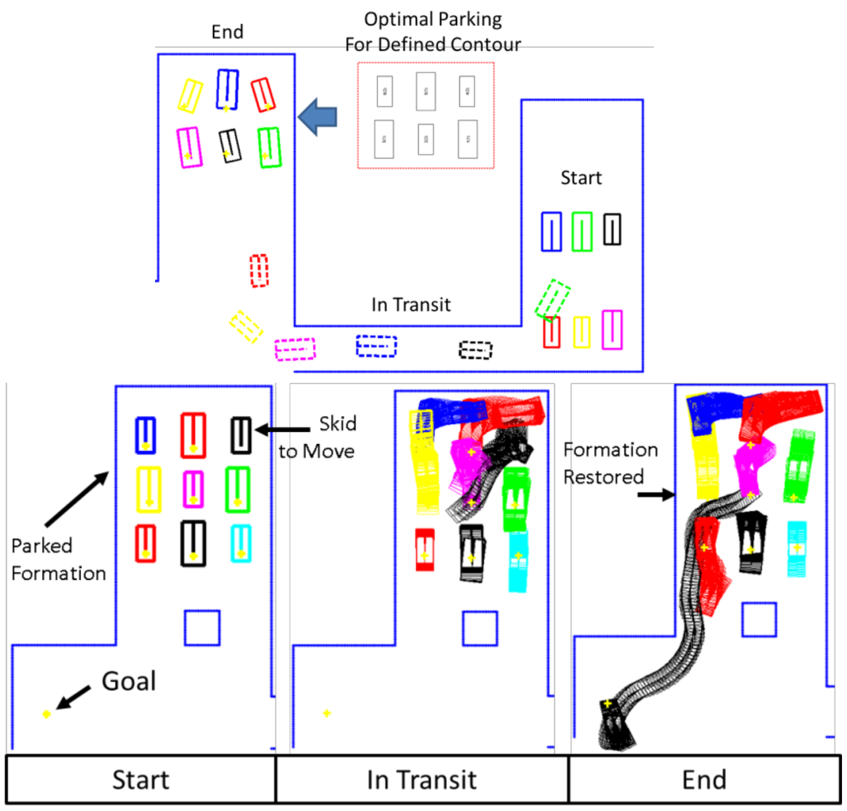

Fig. 19. Top) Simulation of multiple non-holonomic heterogeneous skids moving to a parked formation (dotted lines represent the robot location at time step 200). Bottom) Simulation of a single skid moving through an already parked formation. Positions at each frame are printed to give a visual of the actual trajectories of each skid.

Fig. 14 right. Otherwise, the obstacle is treated as normal and would cause a repulsive force against the skid whose parking spot is occluded or is close to the obstacle.

3.4 Robot & Obstacle Localization

For hardware tests, currently positions of the robots and obstacles are extracted from images obtained from a top down camera. A background subtraction method is used to extract the foreground objects (robots / obstacles) and a pattern matching algorithm is used to identify robot pose at each frame. Each robot has LIDAR sensor that scans 180^o in front of the robot and a series of ultrasonic sensors that cover the back 180^o of the robot. The top down camera also supplies the user with a view from which to select robots and define goal locations. At each frame the pose of the robots are extracted and input into the multi-robot control algorithm. In simulation, each skid knows its goal location, and world location and pose. Obstacle/other skid detection is based on a simulated LIDAR located at the pivot point of the robot with a range of 5m and scan angle of 360^o with a resolution of 0.36^o.

4 Results and Discussion

The algorithms were tested in simulation with $N > 3$ simulated skids and hardware with $N \leq 3$ robots. For the hardware tests, shown in Fig. 18, a kinematic model of the weapons skid was derived and applied to constrain the motion of the robotic research platforms (through software). Many successful scenarios were tested with examples shown in Fig. 18 and **Fig. 19**. In particular, **Fig. 19** bottom shows an example scenario where skids have all parked in a formation and one skid is desired to travel through the formation to a goal location. The nature of the artificial potential field approach allows for this behavior to occur rather naturally. As the skid moves through the formation, the formation itself expands out to let the desired skid through and then contracts back. This is achieved with each skid only knowing its goal location, its current position, and detected distances to surrounding obstacles/other skids. Each skid does not need to communicate directly with other skids to corroborate motion plans as would be needed if more advanced motion planning algorithms were used. This artificial potential field approach lessens the infrastructure needed on the robot for computation and network communication needs. However, a major bottleneck to this approach is the high probability of reaching local minimum, especially when the number of robots or obstacles increases. While many scenarios were successful in our tests, there were a number of scenarios where the final configuration was not achieved due to certain robots reaching local minimums. There are however, many algorithms developed such as the one presented in [15] to handle and escape from these situations.

5 Conclusions and Future Work

This work has presented a simple HMI and artificial potential field approach to parking multiple autonomous non-holonomic skids. The HMI allows the user to select desired robotic skids from a top down view and draw a boundary for which to park the robotic skids. Optimal parking locations within the boundary for each skid are automatically calculated using convex optimization approaches. Simulation and hardware tests have shown successful parking in optimal locations within a boundary as well as moving skids through an already parked formation. While we have not yet implemented the ability to handle local minimums, future work will investigate implementing one of the many escape methods published. We will also consider the use of a hybrid approach using RRT* to escape from the local minimum areas as well. We are also currently investigating the use of simultaneous localization and mapping (SLAM) results for each robot which will be shared amongst each robot and the user interface. This would negate the need for a top-down camera to generate the global position and pose information of the robots however it does increase the computational needs as well as the communication needs of each robot to share their maps of the world.

References

1. N. Cowan, O. Shakerina, R. Vidal and S. Sastry, "Vision-based follow-the-leader," in *IEEE/RJS International Conference on Intelligent Robots and Systems*, 2003.
2. D. P. Scharf, F. Y. Hadaegh and S. R. Ploen, "A survey of space formation flying guidance and control (part 2)," in *American Control Conference*, Boston, MA, 2004.
3. W. Ren and R. W. Beard, "A decentralized scheme for spacecraft formation flying via the virtual structure approach," in *American Control Conference*, Denver, 2003.
4. K. Kondak and G. Hommel, "Compuation of Time Optimal Movements for Autnomous Parking of Non-Holonomic Mobile Platforms," in *International Conference on Robotics and Automation*, Seoul, Korea, 2001.
5. H. Masaki and L. Kangzhi, "Automatic Parking Benchmark Problem: Experimental Comparison of Nonholonomic Control Methods," in *Chinese Control Conference*, Hunan, China, 2007.
6. J. Lawton, R. Beard and B. Young, "A Decentralized Approach to Formation Maneuvers," *IEEE Transactions on Robotics and Automation,* vol. 19, no. 6, pp. 933-941, 2003.
7. Y. Liang and H.-H. Lee, "Decentralized formation control and obstacle avoidance for multiple robots with nonholonomic constraints," in *American Control Conference*, Minneopolis, MN, 2006.
8. G. Elkaim and R. Kelbley, "A Lightweight Formation Control Methodology for a Swarm of Nonholonomic Vehicles," in *IEEE Aerospace Conference*, Big Sky, MT, 2006.
9. S. Ekanayake and P. Pathirana, "Formations of Robotic Swarm: An Artificial Force Based Approach," *International Journal of Advanced Robotic Systems,* vol. 7, no. 3, pp. 173-190, 2010.
10. H. Seki, S. Shibayama, Y. Kamiya and M. Hikizu, "Practical obstacle avoidance using potential field for a nonholonomic mobile robot with rectangular body," in *IEEE International Conference on Emerging Technologies and Factory Automation*, 2008.
11. S. Boyd and L. Vandenberghe, Convex Optimization, Cambridge University Press, 2009.
12. C. Luo, "Novel convex optimization approaches for VLSI floorplanning," University of Waterloo, Waterloo, 2008.
13. "Maximizing minimum distances between points placed in a polygon," [Online]. Available: http://math.stackexchange.com/questions/718733/maximizing-minimum-distance-between-points-placed-in-a-polygon.
14. M. Grant, S. Boyd and Y. Ye, *CVX: Matlab Software for disciplined convex programming,* 2008.
15. A. Kokosy, F.-O. Defaux and W. Perruquetti, "Autonomous navigation of a nonholonomic mobile robot in a environment.," in *IEEE International Workshop on Safety, Security, and Rescue Robotics*, Sendai, Japan, 2008.