Collaboration and Coordination for Multi-Agent Systems

GROUP 2

Alex Hermansson Gianluigi Silvestri

1992-11-18 1994-01-06

grobgeld@kth.se giasil@kth.se

Abstract

Abstract goes here.

1 Introduction

The goal of the assignment is to deal with problems from the Multi-Agent Systems domain, providing efficient and possibly optimal solutions to them.A Multi-Agent System can be thought of as a group of entities that cooperate to reach either a common or an individual goal.Examples of Multi-Agent Systems can be a group of robots that need to coordinate to clean an area in an efficient way, or several self-driving cars that need to drive safely to their destination avoiding to crash with other cars and obstacles.

The problems faced in the assignment can be grouped in two categories: formation control and efficient area coverage.In the formation control problem, a group of agents need to perform particular tasks while keeping a desired formation. For example,if we use a swarm of Unmanned Aerial Vehicles (UAV)s to transport an object, they will have to maintain the same distance among them until they reach a goal destination. To address the formation control problems, we use the virtual structure method presented in [7].In the efficient area coverage, the task of the agents is to cover a given area with constraints such as time and distance traveled. Different variations of the problem are treated, where the agents either have to visit given locations, or they have to make sure to "see" or "sense" the entire area. Examples of these problems are the delivery of food to several houses with a limited number of vehicles, or to perform surveillance of an area using robots equipped with cameras.The area coverage problem can usually be formulated as a Vehicle Routing Problem (VRP), which is a problem known to be NP-Hard [1]. In order to find an optimal or sub-optimal solution in reasonable computational time, it is a common approach to use a meta-heuristic. In this assignment, the meta-heuristic used to solve the VRP is the Genetic Algorithm.

In the assignment, five problems to solve are given:

- **P2 Vehicle Routing Problem**: Given a group of vehicles with start and goal positions, and a set of points of interest to visit, find optimal paths for the vehicles that cover all the points of interest and end up in the goal positions;
- **P3 UAV search or Vacuum Cleaner (Short range VRP)**: Given a group of UAVs with start and goal positions, and an environment to search using a sensor with short range compared to the environment size, find optimal paths for the UAVs that guarantee that the entire environment has been observed;
- **P4 Indoor UGV search (Long range VRP)**: The same as P3, but the sensor range can cover almost the whole environment (but not see through obstacles);
- **P5 Formation along trajectory**: Given a group of agents and a desired formation, the agents have to maintain the same formation while following a given trajectory;
- **P6 Sport formation**: Given a group of agents, desired formation and a point moving in the environment, one agent must always to "cover" the moving point, while the other agents try to keep the formation. The agent that covers the moving point can change, and all the agent must always stay inside the given environment.

1.1 Contribution

The main focus of the assignment was on the study and development of solutions for the problems described above. We adopted a Virtual Structure approach (section 2.1) for P5 and P6, while we solved P2 using a Genetic algorithm (section 2.3). For P3 and P4 we used clustering approaches to formulate the problems as Vehicle Routing Problem (section 2.2), and then solved them as in P2.

1.2 Outline

In section 2, we discuss previous work related to our study and we describe the background of our approaches. We briefly look at different methods for formation keeping such as the virtual structure, we present the vehicle routing problem and the art gallery problem. Also, some background of the genetic algorithm is presented. In section 3, we go into specific detail about our solutions. We provide parameter values and exact description of the genetic algorithm and its variations that were used. Section 4 is a summary of our experimental results together with the results of competing groups and in section 5, we discuss the quality of our outcome and provide ideas of improvement for future work.

2 Related Work

2.1 Multi-Agent Formation Control

There are several approaches to formation control. A common one is the leader-following control scheme, where one agent is assigned a leader role and the rest of the agents try to follow the leader to maintain the prescribed formation. This approach is well described by Egerstedt and Hu in [5]. One drawback is that the leader does not take into account where the rest of the group is, and this may lead to a situation where an agent is left behind and is not able to recover its position in the formation. This could for example happen in the case of a sharp curve if there are constraints on the maximum velocity or curvature of the motion models. Another successful approach is the virtual structure method introduced by Tan and Lewis in [6]. Here, the agents have desired positions in the formation referred to as the "virtual structure". It has advantages over many other methods to treat formation control since it includes formation feedback that makes sure that no agent breaks the formation, but still ensures stability and a clear representation of the dynamics of the system of agents.

The virtual structure, described by the variable $\boldsymbol{\xi}$, works like a blueprint of what the formation should look like, and where the structure is. For N robots in the plane, the coordination variable $\boldsymbol{\xi}$ is parametrized on time by,

$$\boldsymbol{\xi}(t) = (x_c(t), y_c(t), \theta_c(t), \boldsymbol{D}_d(t), \boldsymbol{\Theta}_d(t))^T, \quad \boldsymbol{\xi} \in \mathbb{R}^{3+2N}, \tag{1}$$

where, $(x_c, y_c)^T$ describes a defined center of the virtual structure, θ_c is the orientation of the structure with respect to the origin, $\boldsymbol{D}_d = (D_{1d}, \ldots, D_{Nd})^T$ is a vector of the distances for the agents from the center, and $\boldsymbol{\Theta_d} = (\theta_{1d}, \ldots, \theta_{Nd})^T$ is a vector with the angles for the agents with respect to the center.

In order to keep the formation during maneuvers, the velocity for $\boldsymbol{\xi}$ has to fulfill certain requirements. We want $\dot{\boldsymbol{\xi}} \to 0$ if the robots are far away from their desired positions in the formation. This means that the structure will always wait for the robots to catch up if they lag behind. Also, we want $\dot{\boldsymbol{\xi}} \to v_{max}$, where v_{max} is the maximum velocity of the robots, if all the robots are close to their desired positions.

We denote the locations of the robots with the variable $\boldsymbol{Z} = (\boldsymbol{z}_1^T, \ldots, \boldsymbol{z}_N^T)^T$ and the desired positions with $\boldsymbol{Z}_d = (\boldsymbol{z}_{1d}^T, \ldots, \boldsymbol{z}_{Nd}^T)^T$. Also, the difference between the desired positions and the actual positions is denoted $\tilde{\boldsymbol{Z}} = \boldsymbol{Z} - \boldsymbol{Z}_d$. The velocity of the formation has to be chosen so that it depends on $\tilde{\boldsymbol{Z}}$. Young, Beard, and Kelsey proposes a way to choose the velocity of the virtual structure, in [7], such that the discussed requirements hold. According to their paper, it can be chosen as,

$$\dot{\boldsymbol{\xi}} = -\gamma(\tilde{\boldsymbol{Z}}) K \tanh\left(\frac{1}{K}(\boldsymbol{\xi} - \boldsymbol{\xi}^d)\right), \tag{2}$$

where

$$\gamma(\tilde{\boldsymbol{Z}}) = \frac{1}{K_F \phi(\tilde{\boldsymbol{Z}}) + 1/k_1}, \tag{3}$$

and

$$\phi(\tilde{\boldsymbol{Z}}) = \frac{1}{N} \tilde{\boldsymbol{Z}}^T \tilde{\boldsymbol{Z}}. \tag{4}$$

There are several components of equation (2). The tanh factor forces the velocity to saturate, thus bounds the absolute value to $\max_{\tilde{\boldsymbol{Z}}} \gamma(\tilde{\boldsymbol{Z}}) K$. The γ factor can be seen as a non-linear gain that measures how well the robots keep the formation, and is dependent on the average tracking error $\phi(\tilde{\boldsymbol{Z}})$. From equations (2) and (3) we can see that if the tracking error is large, $\dot{\boldsymbol{\xi}} \to 0$ and if the error is zero $\gamma \to k_1$ so that $\dot{\boldsymbol{\xi}}$ is bounded by $k_1 K$.

The general control architecture can be seen in figure 1. The i:th robot R_i is controlled by the local control block K_i with the signal $\boldsymbol{u}_i$. The signal is produced from inputs $\boldsymbol{y}_i$ and $\boldsymbol{\xi}$. The local control block also outputs the performance variable $\boldsymbol{z}_i$ that is taken as input in the formation control block F along with the supervisor signal $\boldsymbol{\xi}_g$.

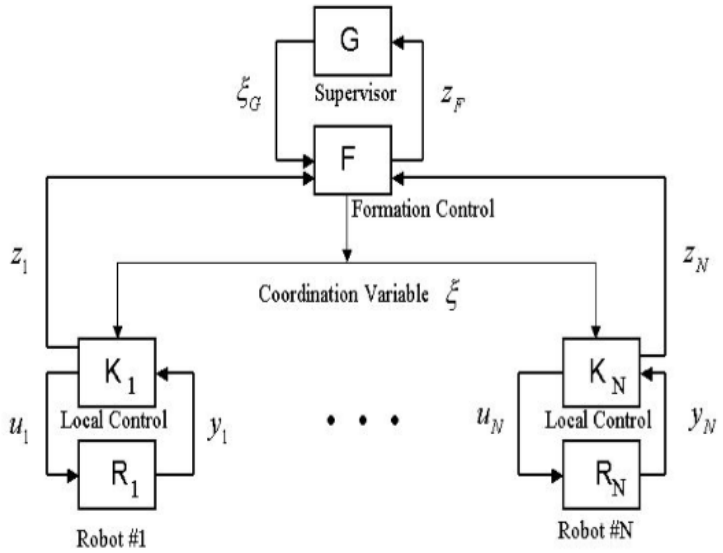

Figure 1: Architecture for robot maneuvers, picture taken from [7].

2.2 Vehicle Routing Problem

The vehicle routing problem (VRP) is a combinatorial optimization problem posed by Dantzig and Ramzer in a paper published in 1959 [4]. The problem is a generalization of the classical traveling salesman problem (TSP) which is concerned with finding the shortest route that passes through a number n of locations. If there are direct links between all locations, there are $\frac{1}{2}n!$ solutions or different routes visiting all places. This very large number of solutions makes it very hard to find the optimal route. In the extension of the TSP to the VRP, there is a number m of vehicles that collectively visit all the locations of interest. In this case, the solution space is even larger. The problem can be defined on a graph (N, A) with the set of nodes N and arcs A. The nodes in the graph are customers in the set $C = \{1, \ldots, n\}$ and the arcs are paths between the customers. Each arc $(i, j) \in A$ has a weight or cost which is, in the simplest case, defined as the distance c_{ij} between the nodes. The cost is symmetric such that $c_{ij} = c_{ji}$ and also $c_{ii} = 0$. The set $V = \{1, \ldots, m\}$ denote the vehicles. We have the decision variable X_{ij}^v defined as,

$$X_{ij}^v = \begin{cases} 1, & \text{if vehicle } v \text{ travels from node } i \text{ to node } j \\ 0, & \text{else} \end{cases} \tag{5}$$

Multiple variations of the VRP exist, but the simplest case of the problem can be mathematically stated as the following optimization problem,

$$\text{minimize} \sum_{v \in V} \sum_{(i,j) \in A} c_{ij} X_{ij}^v \tag{6}$$

subject to the constraints,

$$\sum_{v \in V} \sum_{j \in N} X_{ij}^v = 1, \quad \forall i \in C \tag{7}$$

$$X_{ij}^{v} = \{0, 1\}, \quad \forall (i, j) \in A \text{ and } \forall v \in V \tag{8}$$

$$\sum_{i \in N} X_{ik}^{v} - \sum_{j \in N} X_{kj}^{v} = 0, \quad \forall k \in C \text{ and } \forall v \in V. \tag{9}$$

Equation (6) describes the objective, which is to minimize the total path traveled. The constraint in equation (7) is to make sure that each node in the graph is visited only once. Equation (8) is needed to ensure that equation (5), for the decision variable, holds. Furthermore, equation (9) guarantees that the number of vehicles entering a customer is equal to the number of vehicles leaving it.

2.3 Genetic Algorithm

For the VRP described in section 2.2, a deterministic approach may not be able to find the optimal solution in feasible computational time, especially with relatively large number of agents and points to visit.

One way to overcome this issue is to use a meta-heuristic, that is a procedure able to provide a sufficiently good solution in an acceptable amount of time. A possible meta-heuristic used to solve the VRP is called Genetic Algorithm.

The Genetic Algorithm (GA) is an evolutionary algorithm inspired by the mechanism of natural selection [6], where the best individuals are more likely to survive and their characteristics will be inherited by the next generation. Each individual is called chromosome, and is composed by single units called genes. A chromosome represents a potential solution to the problem (in this case the VRP), and to each one of them is associated a fitness value that indicates the quality of the solution.

In the initialization of the GA, a population of chromosomes is randomly generated. At each iteration, four different operations are executed:

- Parents selection: A number of pairs of chromosomes (parents) reproduce and create two new individuals (children). There are several selection mechanisms, and these are usually random procedures where chromosomes are selected based on their fitness.

- Crossover:In this phase, two chromosomes are generated by combination of the parents' genes. The most common way is to randomly select a cutting point, and the parts on the left of this point are then swapped, generating two new children (Figure 2 left).

- Mutation: After the crossover, the children are subject to mutation with a small probability (usually less than 10%). In a general case, one of the genes is randomly selected and changed (Figure 2 right).

- Survivals selection: A number of individuals is selected from both parents and children to participate to the next generation of the algorithm. The survival selection is based on criteria such as fitness values and age of the individuals (how many iterations an individual survived).

These operations are repeated during the GA, usually for a fixed number of iterations, and the gene with the best fitness found during the algorithm is returned.

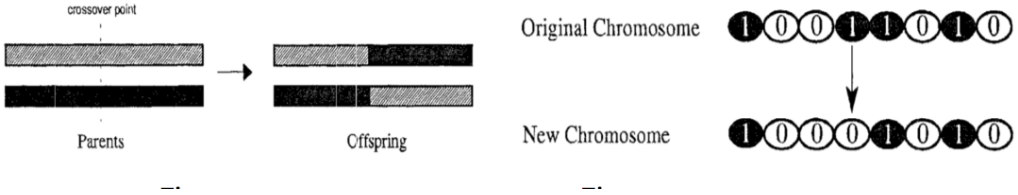

Figure 2: Examples of crossover (left) and mutation (right)

2.4 Art gallery problem

The art gallery problem asks the question of how many guards are needed for surveillance of the gallery, and where they should be placed. The problem was posed in 1973 by Victor Klee, and in 1975 it was solved by Vasek Chvátal in his paper [3] where he proved that, for a simple polygon with n vertices, the number of guards needed are always less than or equal to $n/3$. This is a result known as the art gallery theorem. With this approach it is sufficient to restrict the placement of the guards to vertices, referred to as using vertex guards. For a polygon with h holes, the number of guards needed is proved to be $(n + h)/3$ [2].

3 Our Methods

3.1 P2 - Vehicle Routing Problem

This problem is a variant of the VRP (section 2.2) where each agent has a predefined start and goal position. We are asked to find paths for the agents in order to visit each point of interest only once and minimizing the distance traveled by the agent with the longest path. An example of P2 is shown in Figure 3.

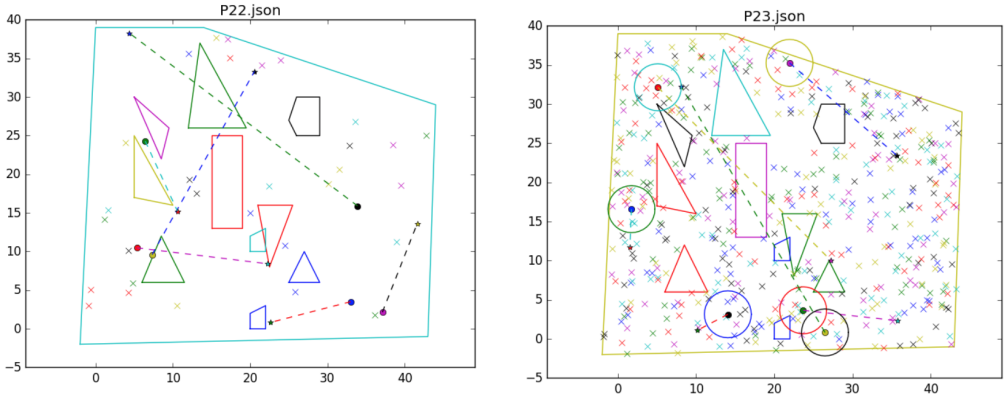

Figure 3: Example of P2 (left) and P3 (right). The small circles are starting positions, while the stars are goal positions and the crosses are points of interest. In P3, the circles around the start positions are the sensors range.

To solve this VRP we use the Genetic Algorithm described in section 2.3, but some modifications are needed.

Let's assume we have N agents and K points of interest. Each chromosome will be a sequence of integers from 1 to $N+K$, where integers from 1 to N are the agents

and numbers from $N+1$ to $N*K$ represent the points of interest. In this way each chromosome contains paths for the agents, as shown in Figure 4.

Figure 4: Example of chromosome with $N = 3$ and $K = 6$: agent 1 will visit points 7-8-4 and then go to its goal position; agent 2 will visit points 3-6-5 and then go to its goal position; agent 3 will go directly to its goal position.

The fitness function is the length of the longest path, and the objective is to minimize it. The crossover used is called *ordered crossover*[DO YOU HAVE NICE REFERENCE?], where the parents are split in two points, and we place the genes in between into the children. The missing genes are filled in the same order as they are in the other parent. Each new generated child can be subject to mutation with a small mutation rate, and two genes are randomly selected and swapped. The parents selection is performed using the so called tournament selection, where each parent is selected by first choosing randomly n chromosomes from the population and then picking the one with lowest fitness among them. At the end of each iteration, all the parents are discarded and all the new children will take part to the next generation.

3.2 P3 - UAV search

In the UAV search, we have a group of agents with start and goal positions. Similarly to P2, several points of interest are given, but in this case is not necessary to visit them. Instead, the points have to be "'seen" by at least one agent, or in other words, to be included into the agent's sensor range with no obstacle between the agent and the point. The sensor range is roughly one order of magnitude smaller than the environment size. An example of environment is shown in Figure 3. The method used to solve this problem is to select a number of points to visit and to use the GA as in P2 (section 3.1), in order to create paths that guarantee that all the points of interest are seen by an agent. To find the points to visit we use the Algorithm 1, where the neighbors are the points of interest seen if an agent is placed on the selected point.

Algorithm 1 Find points to visit

```
Mark all the points of interest as not-seen
Mark as seen the points of interest seen by start and goal positions of the agents
for each point of interest marked not-seen do
    Find neighbors
end for
while at least one point of interest is marked not-seen do
    Select the point with more neighbors
    Add the selected point to the PointsToVisit list
    Mark as seen the selected point and its neighbors
    for each point of interest marked not-seen do
        Update neighbors
    end for
end while
return PointsToVisit list
```

The solution obtained for this problem is not optimal. In fact, the greedy approach doesn't provide the optimal selection of points to visit, and in most cases is not necessary to travel through all the selected points to see all the points of interest.

3.3 P4 - Indoor UGV search

The problem is the same as P3, with the only difference that the sensor range is of similar magnitude as the environment size. The solution that we propose is still to find points to visit an create paths using the GA as in P2. However, due to the extended sensor range, the points to visit are placed in order to guarantee that the whole environment is seen, and thus also all the given points of interest. To do so, we first divide the environment in triangles using the so called Delaunay triangulation [REFERENCE!!!], and then find points to visit with a procedure similar to Algorithm 1. The differences are that, instead of using points of interest, we use the vertices of the obstacles as candidate points to visit, and the neighborhood is the group of triangles entirely seen by the vertex. The vertices that see more triangles are selected greedily until all the triangles are seen by at least one vertex or by one of the start and goal positions of the agents. As in P3, this selection of points to visit is not optimal, and it may not be necessary to visit them all to cover the whole environment. In addition, this method is applicable only if there is no triangle resulting from the triangulation that cannot be entirely contained by the sensor range. If a feasible solution is not found placing the points to visit on the vertices of the obstacles, the candidates can be extended to the points of interest or to all the points inside the environment. DO WE NEED PICS FOR P4???

3.4 P5 - Formation control along trajectory

In this task, a group of robots are asked to travel along a trajectory with a defined formation. The dynamic point motion model was used for the robots, resembling movement similar to a drone's.

The trajectory can be seen as a list of tuples containing position, angle, and time, and represents the supervisor signal in the control architecture in figure 1. In each time step, the supervisor signal $\boldsymbol{\xi}^d$ is provided along with $\tilde{\boldsymbol{Z}}$ to update the velocity of the virtual structure according to equation (2). Also, during this time step, the control signals $\boldsymbol{u}$ for the robots also have to be calculated and broadcast. It was done using a PD controller, giving the signal to the i:th robot as

$$\boldsymbol{u}_i(t) = -k_p\tilde{\boldsymbol{z}}_i(t) - k_d\dot{\tilde{\boldsymbol{z}}}_i(t), \quad \forall i \tag{10}$$

where k_p and k_d are the proportional and differential gains respectively (the minus signs are due to the definition of $\tilde{\boldsymbol{z}}$). This control signal was directly set as the acceleration for the robot. If it exceeded the maximum acceleration a_{max}, it was rescaled to have magnitude a_{max}. For our experiments on P5, both gains were set equal to 10 after some hand tuning. This led to a controller with quick response, but without much oscillatory behavior. Further constants in the virtual structure such as K, K_f, and k_1 were set to 1, 5, and 2.1 respectively. The values were chosen in the same order of magnitude as in [7], with the restriction that $k_1K = v_{max}$ to ensure that the velocity of the virtual structure never exceeded the maximum velocity of the robots, as discussed in section 2.1. PICTURE NEEDED?

3.5 P6 - Sport-like formation control

The sports formation problem is similar to P5, however, in this problem the orientation of the formation is constant and one of the agents always has to cover a mobile goal. The test problem we solved in our experiments simulates a football field were our agents hold a formation such as 4-4-2. In order to hold formation and cover the mobile goal efficiently, we divide the field into a grid and assign cells to the agents suitable to the formation in hand. For the 4-4-2 example, we divided the field into a grid with 3×4 cells as shown in figure 5. The top left and right cells are also assigned to the forwards. To ensure that no agent runs outside of the field, there is a constraint on how far the midpoint can go in horizontal and vertical distance from the midpoint. It can easily be calculated for a given formation.

The supervisor signal, or desired position of the virtual structure $\boldsymbol{\xi}^d$ in this problem is given by the translational vector from the desired agent to the mobile goal (NOT REALLY ACCURATE, PUT EQUATION HERE). Since the formation never changes orientation, this vector is enough to describe the full motion of the virtual structure.

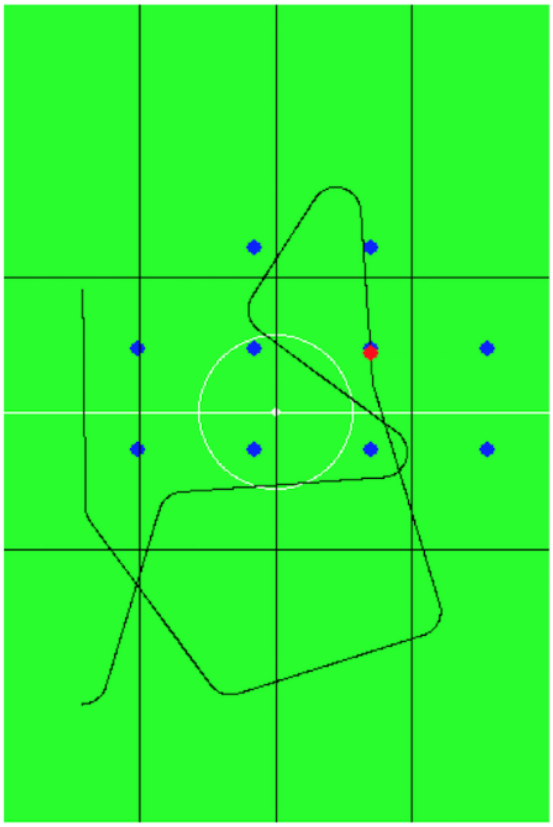

Figure 5: Sports formation with the setup 4-4-2.

4 Experimental Results

5 Discussion and Conclusions

References

[1] Áslaug Sóley Bjarnadóttir. Solving the vehicle routing problem with genetic algorithms, 2004.

[2] Iliana Bjorling-Sachs and Diane L Souvaine. An efficient algorithm for guard placement in polygons with holes. *Discrete & Computational Geometry*, 13(1):77–109, 1995.

[3] Vasek Chvatal. A combinatorial theorem in plane geometry. *Journal of Combinatorial Theory, Series B*, 18(1):39–41, 1975.

[4] George B Dantzig and John H Ramser. The truck dispatching problem. *Management science*, 6(1):80–91, 1959.

[5] Magnus Egerstedt and Xiaoming Hu. Formation constrained multi-agent control. *IEEE transactions on robotics and automation*, 17(6):947–951, 2001.

[6] Kar-Han Tan and M Anthony Lewis. Virtual structures for high-precision cooperative mobile robotic control. In *Intelligent Robots and Systems' 96, IROS*

96, Proceedings of the 1996 IEEE/RSJ International Conference on, volume 1, pages 132–139. IEEE, 1996.

[7] Brett J Young, Randal W Beard, and Jed M Kelsey. A control scheme for improving multi-vehicle formation maneuvers. In *American Control Conference, 2001. Proceedings of the 2001*, volume 2, pages 704–709. IEEE, 2001.