HIERARCHICAL SEARCH STRATEGY FOR A TEAM OF AUTONOMOUS VEHICLES

Jorge Silva [*] **Alberto Speranzon** [***]
João Borges de Sousa [**] **Karl Henrik Johansson** [***]

[*] *Instituto Superior de Engenharia do Porto, Rua Dr. António Bernardino de Almeida N. 431, 4200-072 Porto,Portugal*
[**] *Faculdade de Engenharia da Universidade do Porto, Rua Dr. Roberto Frias, 4200-465 Porto, Portugal*
[***] *Dept. of Signals, Sensors & Systems, Royal Institute of Technology, SE-100 44 Stockholm, Sweden*

Abstract: A multi-vehicle search strategy based on the simplex algorithm is proposed. The strategy is decomposed into a hierarchical scheme with three layers. The upper layer is described by a discrete-event system. The output of this layer is a set of way-points for the vehicles and it is used by the middle layer in order to coordinate the motion of the vehicles. The lower layer drives each vehicle to a way-point. The paper compares two possible discrete coordination strategies: one minimizes the travelled distance of the vehicles and the other avoids their trajectories to cross. Minimization of vehicle intercommunication is also studied.

Keywords: Multi-Robot Systems, Simplex Algorithm, Discrete-Event Systems, Autonomous Underwater Vehicles (AUVs).

1. INTRODUCTION

In this paper we are concerned with a specific multi-vehicle coordination and control problem. Given a scalar field, we desire to coordinate the motions of a group of autonomous vehicles with sensing and very constrained communication capabilities to find its minimum in a given region. In this paper we report our investigations concerning a search strategy based on the fixed size simplex algorithm (Spendley *et al.*, 1962).

The main motivation of this study comes from the PISCIS project, at the Underwater Systems and Technology Laboratory (USTL), Porto University, which offers an operational test bed consisting of two Autonomous Underwater Vehicles (Cruz *et al.*, 2003). Therefore, in this paper most of the results are depicted for a team of two vehicles.

Optimization algorithms have been used as the inspiration for other multi-vehicle search strategies. Bachmayer et al. (Bachmayer and Leonard, 2002) use a pure gradient-based method for scenarios where a vehicle platoon searches the minimum of general convex and smooth scalar fields. However, the oversimplified model considered for the vehicles on that work makes their results unpractical for the vehicles we consider on this work. Burian et al. (Burian *et al.*, 1996) report results, for single vehicle scenario, with several strategies, including combination of the underlying principles of different optimization algorithms, and present illustrative examples using real data, such as depth profiles of a lake.

The main contributions of this paper are the definition of a new motion coordination strategy for the vehicles performing the search operation and a communications protocol for the distributed implementation of the simplex algorithm.

The paper is organized as follows. Section 2.1 gives a brief description of the formulation of the simplex algorithm. Section 2.2 introduces two vehicle allocation strategies. On section 3 we discuss communication between vehicles and present it in the hierarchical framework. Section 4 presents simulation results which illustrate the overall behavior of the system, namely the interaction between the discrete and continuous layers. Section 5 summarizes the conclusions and future work.

2. HIERARCHICAL MODEL

The motivation for employing a team of vehicles is to improve the search operation by sharing the measurements taken by each vehicle. However, as pointed out in the introduction, due to communication constraints, there are limitations on how often measurements can be exchanged between the autonomous vehicles. We therefore propose a control strategy in which decisions only take place at a discrete set of events. In between these events, the vehicles navigate independently.

The control strategy is hierarchical (Varaiya, 2000) with three layers: the upper layer is modelled by a discrete-event system (DES) which communicates to the lower layers locations that must be sampled. The Vehicle Allocation layer assigns to the available vehicles the task of sampling those points, according to some predefined strategy. The lower layer is composed by the supervisors and continuous-time regulation laws that ensure correct guidance of the vehicles to the desired points. In the following we describe the layers in detail.

2.1 DES layer: simplex algorithm

The search strategy described in this paper is based on the simplex algorithm. In this section we formalize the simplex algorithm as a Discrete Event System.

The simplex algorithm is a direct search method used in many practical optimization problems. It is usually applied in situations where the cost of function evaluation is high and gradient calculation is difficult, as happens in this case. The cost of function evaluation can be related to the cost of measurement's broadcast. The algorithm is useful to improve an initial estimate of the solution with few function evaluations. Its simplicity and robustness properties (Nelder and Mead, 1965; Lagarias *et al.*, 1998) make it an interesting algorithm for minimum search applications with multiple vehicles. Notice that the widely used gradient based methods cannot cope with the existence of noise in the field. Moreover, the main objective in this kind of application is not an algorithm which converges in few iterations but one which enhances the synergy between the vehicles given the problem constraints. The sampling sequence determines the time taken to reach the minimum since it depends on the total distance travelled by the vehicles. This observation rules out optimization methods which perform well on a pure mathematical analysis but which would lead to inefficient sampling sequences (for example, "simulated annealing" (Burian *et al.*, 1996)).

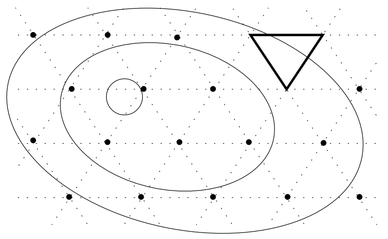


Fig. 1. A triangular grid over a two-dimensional scalar field depicted by its level curves. The solid line triangle illustrates the state $\mathbf{z}$ of the discrete-event system evolving on the grid.

Thus, our objective is to use the simplex algorithm to progress towards the minimum and to get as close as possible to it. In order to improve the estimate of the solution obtained with the simplex algorithm we can resort to other kind of search strategies, which are topic of future work.

An informal description of a multi-vehicle search strategy based on the simplex algorithm was first presented in (Sousa and Pereira, 2002). In (Speranzon *et al.*, 2004), a formal description of the system architecture and a Discrete Event Systems formulation of the simplex are presented. We follow that formulation in this paper.

Define a field through a scalar-valued map $F : \Omega \to \mathbb{R}$. Let $\mathbf{p}_i \in \Omega$, $i = 1, 2, \dots$ be successive key positions visited by the vehicles. The fixed-size simplex coordination scheme is a discrete-event system $D = (\Sigma, Z, W, \xi, \mathbf{z}_0)$. For a reference on discrete-event systems notation see (Cassandras and Lafortune, 1999). The state $\mathbf{z}$ is a triangle of neighboring grid points (See Figure 1)

$$\mathbf{z} = \{\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3\}$$

Let the input set W be a subset of $\mathbb{R}^3$ with $\mathbf{w}^T = (F(\mathbf{p}_1), F(\mathbf{p}_2), F(\mathbf{p}_3)) \in W$. The transition function ξ is defined as

$$\xi(e, \mathbf{z}, \mathbf{w}) = \mathbf{z}\mathbf{P}_{\mathbf{w}}\mathbf{S}$$

where $\mathbf{P}_{\mathbf{w}}$ is the permutation matrix

$$\mathbf{P_w} = \begin{cases} \begin{pmatrix}\mathbf{e}_3 & \mathbf{e}_2 & \mathbf{e}_1\end{pmatrix}, & \text{if } w_1 > \max(w_2, w_3) \\ \begin{pmatrix}\mathbf{e}_1 & \mathbf{e}_3 & \mathbf{e}_2\end{pmatrix}, & \text{if } w_2 > \max(w_1, w_3) \\ \begin{pmatrix}\mathbf{e}_1 & \mathbf{e}_2 & \mathbf{e}_3\end{pmatrix}, & \text{otherwise} \end{cases}$$

with $\mathbf{e}_i \in \mathbb{R}^3$ being the unit vectors, and

$$\mathbf{S} = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & -1 \end{pmatrix}$$

The transition function hence updates the simplex $\mathbf{z}$ by picking the point $\mathbf{p}$, say, in $\mathbf{z}$ with largest value $F(\mathbf{p})$, and then reflects it in the axis between the other two points of $\mathbf{z}$. The new point replaces the reflected one and forms the new simplex $\mathbf{z}^+ = \xi(e, \mathbf{z}, \mathbf{w})$. In what follows, the simplified notation $\mathbf{z}^+ = \mathcal{S}(\mathbf{z})$ is used for the discrete update and $\mathbf{z}(k)$ denotes the discrete state after event e_k, $k = 0, 1, \ldots$. The discrete evolution terminates at step $k^* \geq 2$ if $\mathbf{z}(k^*) = \mathbf{z}(k^* - 2)$. Since the algorithm is deterministic, it follows that a continuation beyond step k^* would lead to an oscillation between the two discrete states: $\mathbf{z}(k^*)$ and $\mathcal{S}(\mathbf{z}(k^*))$. We call the union of these a *fixed point* of the discrete evolution since $\mathbf{z}(k^*) = \mathcal{S}(\mathcal{S}(\mathbf{z}(k^*)))$ and denote it $\chi = \mathbf{z}(k^*) \cup \mathcal{S}(\mathbf{z}(k^*))$.

Remark 1. The vertex which replaces the worst vertex (i.e., the vertex $\mathbf{p} \in \mathbf{z}$ that solves $\max_{\mathbf{p}\in\mathbf{z}} F(\mathbf{p})$) from the simplex $\mathbf{z}$ will be the third element of $\mathbf{z}^+$.

2.2 Vehicle allocation layer

The simplex algorithm, as described above, returns points to be sampled. However, it is still necessary to choose which vehicle will travel to which point in order to sample it. In this section we present two vehicle-to-sampling point allocation strategies: non-crossing and greedy.

In practice, the first simplex will be predefined on both vehicles with a random initial vertex allocated to each vehicle. Let's assume that vehicle 1 (V_1) is at vertex A and vehicle 2 (V_2) is at vertex B. C is the unvisited vertex.

(1) V_2 samples the value at its initial vertex B and goes to the unvisited vertex C.
(2) V_1 compares $F(A)$ with $F(B)$. If $F(A)$ is worse than $F(B)$, V_1 goes to that vertex, making it its current vertex.
(3) Eventually, V_2 reaches C. If $F(C)$ is worse than $F(B)$, special care has to be taken: if $F(B)$ is worse than $F(A)$, V_2 calculates the new vertex and goes to there (the algorithm advances one iteration). If $F(A)$ is worse than $F(B)$, V_2 will wait the result from V_1 telling it whether $F(C)$ is worse or not than $F(A)$. If $F(C)$ is worse than $F(A)$, V_2 will also calculate and advance to next vertex.

This concludes the initialization phase, with the vehicles at the two best vertexes of the current simplex.

Starting from $\mathbf{z}(k)$ and assuming that this is not the fixed point, it is always possible to generate the two following simplices, $\mathbf{z}(k+1)$ and $\mathbf{z}(k+2)$, without knowledge of[1] $F(\mathbf{z_3}(k+1))$.

The set of vertexes $\mathbf{z}(k+1)$ is trivially obtained by application of the simplex algorithm. If the vehicle is at the fixed point then $\mathbf{z_3}(k+1)$ will be the worst vertex of $\mathbf{z}(k+1)$, p_w, and the algorithm will stop as soon as that vertex is sampled. When that is not the case, p_w will have to be one of the two other vertexes, whose values are already known, and hence its determination is trivial. In this case, it is possible to "predict" $\mathbf{z_3}(k+2)$ even without completing the current iteration, i.e., without waiting for the acquisition of the value at $\mathbf{z_3}(k+1)$, just by reflecting p_w through the remaining vertexes of $\mathbf{z}(k+1)$. Therefore, at each step of our algorithm there are two new candidate points to be sampled, which may be allocated to each vehicle. It is possible to have one vehicle travelling to the point required to complete the current iteration and the other travelling to the point required to perform the next iteration.

We define the permutation matrixes $\mathbf{P}^+_\mathbf{w}$ and $\mathbf{P}^{++}_\mathbf{w}$ as

$$\mathbf{P}^+_\mathbf{w} = \begin{cases} \begin{pmatrix}\mathbf{e}_1 & \mathbf{e}_3 & \mathbf{e}_2\end{pmatrix}, & \text{if } w_2 > \max(w_1, w_2) \\ \begin{pmatrix}\mathbf{e}_2 & \mathbf{e}_3 & \mathbf{e}_1\end{pmatrix}, & \text{otherwise} \end{cases}$$

$$\mathbf{P}^{++}_\mathbf{w} = \begin{pmatrix}\mathbf{e}_1 & \mathbf{e}_3 & \mathbf{e}_2\end{pmatrix}$$

and

$$\mathcal{S}^+(\mathbf{z}) = \mathbf{z}\mathbf{P}^+_\mathbf{w}\mathbf{S}$$
$$\mathcal{S}^{++}(\mathbf{z}) = \mathbf{z}\mathbf{P}^{++}_\mathbf{w}\mathbf{S}$$

Thus, at each step of the simplex algorithm we will also perform the following calculations:

$$\hat{\mathbf{z}}(k+2) := \mathcal{S}^+(\mathbf{z}(k+1))$$
$$\hat{\mathbf{z}}(k+3) := \mathcal{S}^{++}(\mathbf{z}(k+2))$$

The motivation for the non-crossing allocation strategy comes from the requirement to minimize the probability of collisions. The *non-crossing* vehicle allocation strategy is defined as follows: the vehicle with the most recently visited worst vertex, the *exploring vehicle* will move to the next vertex to be sampled. The other vehicle, the *waiting vehicle*, will either stop, if its dynamics allows it, or move towards $\hat{\mathbf{z}}_\mathbf{3}(k+2)$ at lowest possible speed.

In the simulations we used as reference p_b, the point between $\hat{\mathbf{z}}_\mathbf{3}(k+2)$ and $\hat{\mathbf{z}}_\mathbf{3}(k+3)$. The choice

[1] We denote with $\mathbf{z}_i(k)$ the i-th element of the set of vertexes $\mathbf{z}(k)$.

of p_b is made based on the observation that the only candidate vertexes for the waiting vehicle in the following two iterations are those two points. It can be seen that after *exploring vehicle* samples $\mathbf{z}_3(k)$, this vertex will be found either as:

- The worst vertex, in which case the algorithm stops.
- The new best vertex, in which case the former waiting vehicle starts travelling to $\hat{\mathbf{z}}_\mathbf{3}(k+2)$ as exploring vehicle.
- The second best vertex. In this case, as is implicit in the strategy, both vehicles keep their roles. The exploring vehicle starts travelling to $\hat{\mathbf{z}}_\mathbf{3}(k+2)$ and the next possible point to be sampled by the waiting vehicle will be $\hat{\mathbf{z}}_\mathbf{3}(k+3)$.

The *greedy strategy* is defined as follows: given $\mathbf{z}_\mathbf{3}(k+1)$, $\hat{\mathbf{z}}_\mathbf{3}(k+2)$ and the current position of the vehicles, the vertexes are allocated to each vehicle in order to minimize the sum of the travel distance of both vehicles from their current position to the destination vertexes. The underlying idea of this strategy is to allow the vehicles to advance as fast as possible by taking advantage of the anticipated simplex reflection $\hat{\mathbf{z}}(k+2)$.

2.3 Continuous dynamics

The continuous layer represents the dynamics of the vehicles and the continuous-time control algorithms. Each vehicle $i = 1, \dots, n$ is described by a nonlinear control system

$$\dot{\mathbf{x}}_i = f_i(\mathbf{x}_i, \mathbf{u}_i), \qquad \mathbf{u}_i \in U_i$$

where $f_i : \Omega \times U^i \to \Omega$ defines the dynamics of the individual vehicles with continuous state $\mathbf{x}_i$ and admissible continuous controls in U_i. The control $\mathbf{u}_i$ is a state feedback that depends on both the continuous state and a waypoint $\mathbf{p}_i$ determined by the supervisor, i.e.,

$$\mathbf{u}_i = \mathbf{u}_i(\mathbf{x}_i, \mathbf{p}_i)$$

When vehicle i is sufficiently close to $\mathbf{p}_i$, the field measurement is communicated to the upper layers.

3. VEHICLE INTERCOMMUNICATION

Since we are dealing with autonomous vehicles we desire to minimize the communication between the vehicles, in order to limit the power consumption associated to communications. The execution of the simplex algorithm requires, at each event instance, the two vehicles to know which vertex is to be reflected, i.e. the worst vertex, and from there to compute the next vertex that has to be sampled. Thus the issue is about the possibility to infer on which is this vertex without communicating measurements.

In this paper we consider the following communications scheme. Each time V_2 samples a new point it broadcasts the sampled value. V_1 never broadcasts its sampled points. Hence, by this manner, V_1 is able to take decisions at each event, since it centralizes all data. As we'll show below the communication bandwidth is reduced by a factor of almost two when compared to the case when both vehicles broadcast their sampled values. This factor is not exactly two because V_2 has to somehow decide which point it should sample next. That can be done by communicating the relative order of the three vertexes of the current simplex from V_1 to V_2. This would amount to six possible cases (the permutations of the three vertexes). However, the possible cases can be further reduced due to the restrictions imposed by the trajectory coordination strategy as it will be shown below. The following definitions will be based on the non-crossing allocation strategy.

Remark 2. Since after the initialization phase, both vehicles know that the remaining vertex of the current simplex is the worst, V_2 can calculate, at iteration k, $\mathbf{z}_3(k+1)$ and $\mathbf{z}_3(k+2)$ without knowledge of the function value of the vertexes visited by V_1 just by using the relative order between its current vertex and V_1' last sampled vertex.

Thus, the protocol is completed as follows. Whenever V_1 samples or receives a new vertex's value, $\mathbf{z}_3(k)$, it is able to decide between the following cases:

(1) The new vertex is the worst one. The algorithm stops.
(2) V_1's last visited vertex is the best vertex. In this case, V_1 will behave as the waiting vehicle and will signal V_2 to sample the new reflected vertex, $\mathbf{z}_3(k+1)$, with a goto command.
(3) Otherwise, it means that V_2's last visited vertex corresponds to the best vertex and that it should become (or remain as) the waiting vehicle. V_1 emits the corresponding signal (hold command) and starts travelling to $\mathbf{z}_3(k+1)$. Upon receiving this signal, V_2 will calculate and start pointing to $\mathbf{z}_3(k+2)$.

This shows that three different codes will be enough to indicate V_2 which way to follow, without transmission of any of V_1's sampled values. Thus V_1 has to transmit only 2 bit to encode one of the two possibilities. Compare with a full broadcast of information we reduce the total communication from 32 bits (assuming data samples of 16 bits) to 18 bits thus reducing to almost half the total number of bits transmitted.

Figure 2 illustrates the protocol in the context of the system architecture. In order to avoid figure cluttering we avoided the obvious state designations, with the exception of the automata from the upper layer of V_2: state 1 means that V_2 is in the *exploring vehicle* role, in response to a goto command; state 2 corresponds to the case when V_1 is in the *waiting vehicle* role; the automata transits to state 3 after V_2 transmits a new sample to V_1 and remains in that state until a response is received. This layer receives the commands goto and hold without the reference points. For each transition (again, this is not represented in the figure) labelled with goto or hold, the simplex algorithm is executed and the new simplex is calculated and sent to the allocation layer. At that time, the allocation layer adds the new vertex to the received command and transmits this data to the lower layer.

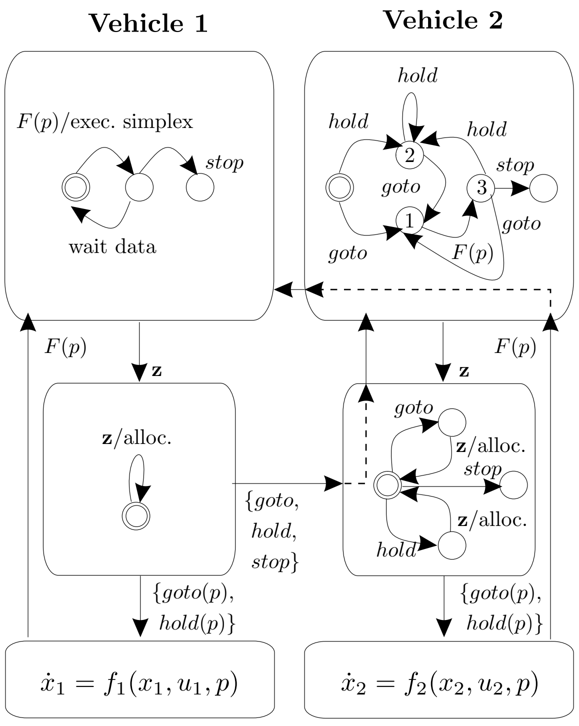


Fig. 2. Hierarchical Architecture.

4. SIMULATION RESULTS

Table 1 shows simulations results for both strategies considering a team of two vehicles with velocity limited to a maximum of 2 m/s. We considered a intermediate case, the greedy strategy with velocity limited to 1.6 m/s. The last parameter (actuation effort) is based on approximate values of our reference vehicles' (Cruz *et al.*, 2003) dynamic coefficients and does not intend to be an absolute estimate of the power consumption. The simulation consisted of 360 runs of the simplex based search, with both allocation strategies, using the test function $F(x, y) = x^2 + 2y^2$, as an example of a local approximation of a phenomena showing a slight conditioning. In each run the initial simplex has a random location (with a distance of 800 meters from the minimum) and orientation.

There is a trade-off in setting the size of the simplex in the simplex optimization algorithm: on one hand, to ensure better accuracy it should be made as small as possible because the size of the simplex determines a discretization of the space; on the other hand, the size of the simplex is limited below by the dynamic behavior of the vehicle. For non-holonomic vehicles, the trajectories generated by the algorithm may become impractical if the grid size is too small.

Table 1. Simulation Results (Average/Standard Deviation)

Allocation Strategy	Non-crossing	Greedy (1.6 m/s)	Greedy (2 m/s)
Completion time (s)	699/46	673/146	532/110
Total travelled distance (m)	1818/121	2005/299	1984/279
Actuation Effort (J)	55k/4k	48K/4k	74k/6k

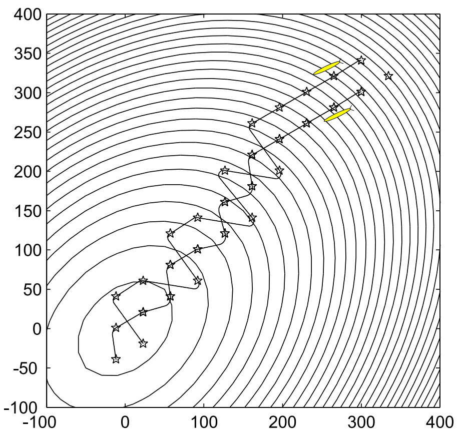
Depending on the initial point, the greedy strategy may generate trajectories similar to the non-crossing strategy or give rise to longer ones. Figures 3(a) and 3(b) illustrate a scenario where the strategies lead to very different resulting trajectories.

The average results show that, in spite of the odd trajectories, in general the greedy strategy leads to shorter completion times and with more efficient power usage. This is explained by the fact that with the non-crossing strategy there's always one vehicle stopped or moving at very low speed, waiting for the results of the other one, while in the greedy strategy both vehicles are always moving to a new vertex.

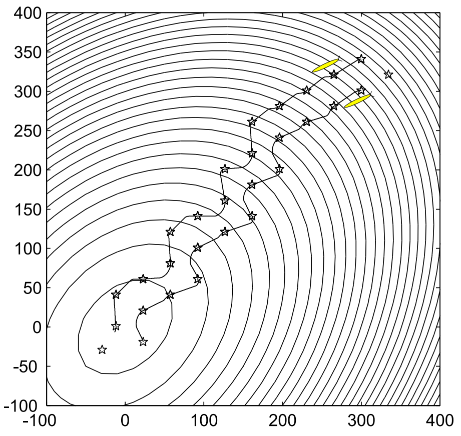
Simulations were also performed for other valley shaped functions with different levels of superimposed gaussian noise and we observed a direct correspondence between the level of noise and the final distance to the minimum, which confirms the expected robustness of the simplex algorithm.

5. CONCLUSIONS

This work shows one possible way of taking advantage of the availability of multiple vehicles on a search operation on a static field. It is easy to see that the results obtained with the team of two vehicles surpass the ones that would be obtained for a single vehicle using the same simplex based



(a) Trajectories with greedy vehicle's allocation strategy.



(b) Trajectories with non-crossing vehicle's allocation strategy.

Fig. 3. Comparison of vehicles' trajectories obtained with both allocation strategy

method. The employment of more vehicles, which, for instance, would be interesting for time varying fields, is left as topic of future work.

Our approach explore the advantages of the simplex method in what concerns easiness of distribution, at expenses of suboptimal vehicles' trajectories. Notice that the vehicles must displace through the discrete points of the grid giving rise to sinuous trajectories. However, it is the same discretization of the state space which leads to the definition of the lean communication protocol, which allows the employment of a very simple communication scheme in one of the directions, eventually with simplified hardware and reduced power consumption which is of major importance on autonomous systems. If symmetric communication poses no special problem, we think (based on our preliminary studies) that better results may be obtained using gradient based techniques together with filtering. We intend to pursue this idea on future work.

REFERENCES

Bachmayer, R. and N. E. Leonard (2002). Vehicle networks for gradient descent in a sampled environment. In: *Proc. 41st IEEE Conference on Decision and Control*. Las Vegas, NV, USA.

Burian, E., D. Yoerger, A. Bradley and H. Singh (1996). Gradient search with autonomous underwater vehicles using scalar measurements. In: *Proc. of the IEEE Symposium on Autonomous Underwater Vehicle Technology*. Monterey, CA, USA. pp. 86–89.

Cassandras, C.G. and S. Lafortune (1999). *Introduction to Discrete Event Systems*. Kluwer Academic.

Cruz, N., A. Matos, J.B. Sousa, F.L. Pereira, J. E. Silva, E.P. Silva, J. Coimbra and E.B. Dias (2003). Operations with multiple autonomous underwater vehicles: the piscis project. In: *Proc. of the Second Annual Symposium on Autonomous Intelligent Networks and Systems*. Menlo Park, CA, USA.

Lagarias, J. C., J. A. Reeds, M. H. Wright and P. E. Wright (1998). Convergence properties of the nedler-mead simplex in low dimensions. *SIAM Journal of Optimization* **9**(1), 112–147.

Nelder, J.A. and R. Mead (1965). A simplex method for function minimization. *Computing Journal* **7**, 308–313.

Sousa, J. B. and F. L. Pereira (2002). On coordinated control strategies for networked dynamic control systems - an application to auvs. In: *Proc. of the Fifthteenth International Symposium of Mathematical Theory of Networks and Systems (MTNS)*. Notre-Dame, USA.

Spendley, W., G.R. Hext and F.R. Himsworth (1962). Sequential applications of simplex designs in optimization and evolutionary operation. *Technometrics* **4**, 441–461.

Speranzon, A., J. E. Silva, J. B. Sousa and K. H. Johansson (2004). On collaborative optimization for a team of autonomous underwater vehicles. In: *Submitted to CDC2004*.

Varaiya, P. (2000). A question about hierarchical systems. In: *System Theory: modeling, analysis and control* (In T. Djaferis and I. Schick, Eds.). Kluwer.