Cooperative control through objective achievement

Alessandro Farinelli [a] Hikari Fujii [b] Nanase Tomoyasu [b] Masaki Takahashi [b]
Antonio D'Angelo [c] Enrico Pagello [d]

[a] *Dept. of Computer Science, University of Verona, I-37134, Verona (Italy)*
[b] *Dept. of System Design Engineering, Graduate School of Science and Technology, 3-14-1 Hiyoshi, Kohoku-ku, Yokohama, 223-8522 Japan*
[c] *Dept. of Mathematics and Computer Science, Via delle Scienze 206, I-33100 Udine (Italy)*
[d] *Dept. of Electronics and Engineering, Via Gradenigo 6, I-35131 Padova (Italy)*

Abstract

Cooperative control is a key issue for multirobot systems in many practical applications. In this paper, we address the problem of coordinating a set of mobile robots in the RoboCup soccer middle size league. We show how the coordination problem that we face can be cast as a specific *coalition formation problem* and we propose a distributed algorithm to efficiently solve it. Our approach is based on the distributed computation of a measure of satisfaction (called *Agent Satisfaction*) that each agent computes for each task. We detail how, each agent computes the *Agent Satisfaction* by acquiring sensor perceptions through an omnidirectional vision system, extracting aggregated information from the acquired perception, and integrating such information with the ones communicated by the team mates. We empirically validate our approach in a simulated scenario and within the RoboCup competitions. The experiments in the simulated scenario allow us to analyse the behaviour of the algorithm in different situations, while the use of the algorithm in the real competitions validates the applicability of our approach to robotic platforms involved in a dynamic and complex scenario.

Key words: multirobot system, coordination, task assignment, RoboCup

1. Introduction

The growing interest to develop colonies of robots engaged in complex tasks like search and rescue, monitoring environmental phenomena and surveillance in security applications, has caused an increasing interest towards coordination approaches which can provide flexible and reliable solutions. In fact, the coordination of the robotic platforms' activities can increase both the efficiency of the global task execution and the robustness of the system to individual robot failures.

However, devising flexible and effective coordination methods for multirobot systems is a very complex and challenging task. Coordination in these domains is particularly difficult because it requires the solution to be distributed among the robots to enhance robustness and avoid the existence of a central point of failure; moreover, the environment that robots face is highly dynamic and unpredictable, therefore the coordination method should be able to react to unexpected changes and provides good quality solutions minimising the reaction time; finally, robotic platforms interact with the world through sensors and actuators which are inherently noisy and inaccurate; this results in uncertainty both in perceptions and actions.

Agent-Based coordination techniques are widely used to achieve cooperative behaviours in distributed settings, and in particular, here we focus on coalition formation (17). In coalition formation, a set of robots must cooperate to accomplish a set of tasks (or roles). Each robot can execute one task at the time but the robots can form coalitions to cooperate on specific tasks. Coalitions can perform tasks better (e.g., faster or in a more reliable way) than single robots, and the quality of the execution of a specific task [1] depends both on the individual capabilities that each robot has for that task, and on how the capabilities can be combined together. Several approaches have been studied to address the coalition formation problem (e.g., (17; 16)) which are able to compute the optimal solution. However, coalition formation is known to be an NP-Hard problem and even if consistent improvements have been achieved on the computation time of the optimal solution, such approaches have still limited applicability in dynamic scenarios where the value associated with a coalition changes very rapidly over time.

In this paper we present an approach to coalition formation explicitly targeted for dynamic uncertain environments. The approach is based on (6) and computes in a distributed way, a measure of satisfaction called *Agent Satisfaction*. The *Agent Satisfaction* represents the level of satisfaction that each agent has for a task being executed by a set of agents. Each agent computes the *Agent Satisfaction* by acquiring sensory perceptions (which in our specific case are images coming from an omnidirectional vision system), extracting aggregated information and integrating them with information transmitted by other team mates.

The proposed coordination approach is explicitly designed for scenarios where tasks to be executed can be ranked according to priorities. By exploiting this assumption the method is able to address in an efficient way the coalition formation problem and is of practical use in dynamic environments, where agent capabilities to execute the tasks can rapidly change over time.

Specifically, we apply our approach to the RoboCup soccer scenario, and in particular to the middle size league. In the RoboCup soccer middle size league, two teams of robots play a soccer game against each other. This scenario is particularly interesting as a benchmark for coordination algorithms, because it is highly dynamic and the game evolution is highly unpredictable. Moreover, the RoboCup competitions have become, over the years, a very important event attracting hundreds of researchers from all the world, to compete with robotic systems in a specific scenario. Therefore they represent a unique testbed for comparing different systems and approaches to various robotic problems and specifically coordination.

We validated our approach by means of experiments both in a simulated soccer scenario and in the RoboCup competitions. The simulated experiments allow us to analyse the behaviour of our algorithm in a controlled setting and under different operative conditions, while the data collected from the RoboCup competitions validate the applicability of our approach in an autonomous multirobot system. The coordination method described here was used by the EIGEN team during the RoboCup competitions since 2004 and the effectiveness of the coordination method is confirmed by the excellent results that the team achieved during the competitions (first place in 2004 and 2005, third place in 2006 and second place in 2007).

The rest of the paper is organised as follow: section 2 details a formalisation of the coalition problem that we address, while section 3 introduces and discusses the cooperative control method we propose. Section 4 shows how the aggregated information needed by the cooperative control method are extracted from the robot's perceptions. Section 5 specifies our approach to the RoboCup scenario and presents the experimental results from the simulated environment, while Section 6 presents the EIGEN team control architecture and the validation of the systems during the RoboCup competitions. Section 7 discusses relevant previous work and finally, Section 8 concludes the paper.

2. Problem Formalisation

In this section we detail a formalisation of the cooperative control problem that we face. We have a set of robots that must be assigned to a set of roles (or tasks), and while each robot can assume only one role, it can be beneficial for robots to assume the same role. For example, when considering the RoboCup soccer scenario, a *defence* role might require two robots to completely block the opponent and thus prevent it from shooting.

This problem can be naturally formalised as a *coalition formation* problem (17). In coalition formation robots have different capabilities to perform each task, and when they cooperate on the same

[1] This is usually called the *value* of the coalition for that task.

task, the utility they gain is a function of their capabilities. The output of the coalition formation problem is a partition of the robot set into coalitions that are allocated to a specific task. The objective is to maximise the sum of the utility of the coalition assignment. The general problem of coalition formation can be cast as an instance of the Set Partitioning Problem, which is known to be NP-hard (9).

However, in our scenario, we face a specific version for the coalition formation problem. In particular, we can assume that roles have a predefined priority. For example, in the RoboCup soccer domain that we consider here, the role priority is usually defined by a game strategic layer. The strategic layer assigns priorities to tasks according to contextual information, e.g., if the team is winning the strategy layer might assign high priority to the defence task, in order to maintain the current result and win the match (see sect. 5 for further details). Suppose a *Defence* schema is used, where three roles can be performed: *Defence*, *Support* and *Offence*. Furthermore, suppose that in this schema *Defence* has priority over all the other roles, and *Support* has priority over *Offence*. The priority of roles, entails that the value that the team obtains by performing the *Defence* role is higher than the value that they can obtain by performing any combination of all the other roles. This concept of role priority is very natural for the RoboCup domain and was already used in a previous work on coordination in this scenario (10).

More precisely, we can formalise our problem as following: we have a set of robotic agents $\mathcal{R} = \{R_1, ..., R_n\}$ and a set of tasks (or roles) $\mathcal{T} = \{t_1, ..., t_m\}$. Each robotic agent R_i has capabilities to perform each task represented by a vector $S_i = \langle s_i^1, ..., s_i^m \rangle$ where $s_i^j \in \Re$ represents the level of performance that R_i can achieve when allocated to role t_j. Each task t_j has a desired achievement level $l_j \in \Re$ that needs to be reached by the agents accomplishing the task. The level of achievement of a task represents an objective of the whole system and will be called therefore *System Objective* in the following. Considering this interpretation of the achievement level, each of the s_i^j, can then be interpreted as the level of satisfaction that the agent will obtain if it is allocated to the task or *Self Evaluation* of the agent. In other words, the *Self Evaluation* is an estimation that each agents computes of its capability to perform a task.

A coalition C is a set of agents and thus $C \subseteq \mathcal{P}(\mathcal{R})$, where $\mathcal{P}(\mathcal{R})$ is the powerset of $\mathcal{R}$. We indicate with C_j the coalition of agents assigned to task t_j and the set $\mathcal{C} = \{C_1, ..., C_m\}$ represents the set of coalitions assigned to all tasks; on the set $\mathcal{C}$ the following properties hold: i) $C_i \bigcap C_j = \emptyset \ \forall i, j \mid i \neq j$ ii) $\bigcup_{i=1}^{i=m} C_i = \mathcal{R}$; in other words, the set $\mathcal{C}$ is a partition of $\mathcal{R}$. We define a function $F : \mathcal{P}(\mathcal{R}) \times T \Rightarrow \Re$, and $F(C, t_i)$ represents the amount of work that agents in coalition C can perform for task t_i. This measure is an aggregation of the s_k^i of all agents R_k in coalition C when the coalition works on task t_i. The aggregation can be any function, for example in the RoboCup soccer domain is a summation as will be detailed in Section 5.2. We represent with $V_i(C)$ the utility that the system can gain when a coalition successfully accomplish a task t_i, i.e. when $F(C, t_i) \geq l_i$. If a coalition C cannot perform the required workload for task t_i then $V_i(C) = 0$, otherwise $V_i(C) = v_i$, where $v_i \in \Re^+$. To model the priority among tasks, we assume that $v_i \gg v_{i+1}$ $i = 1, \cdots, m-1$. And in particular, we assume that accomplishing a task of higher priority is always better than executing any combination of lower priority tasks. The value $V_i(C)$ represents an aggregation of the individual agents' self evaluations, and indicates the total level of satisfaction that the agents inside the coalition have for task t_i, we call this value *Agent Satisfaction*.

Given the above definitions our objective is then:

$$\arg\max_{\mathcal{C}} \sum_{i=1}^{m} V_i(C_i) \tag{1}$$

Coalition formation is usually a one-shot problem where coalitions values are known in advance, and once coalitions are formed and allocated to tasks, robots will simply carry on their tasks. However, in our domain robots have to deal with a more complex setting. In fact, since we have an opponent playing against our team, the situation changes very rapidly; this means that the *Self Evaluation* that each robotic agent computes for each task will change over time as well as the value of coalitions, the priorities of tasks and the *System Objective*. In particular, priorities of tasks and the *System Objective* can change due to many domain specific issues. For example, role priorities can change because the context of the game changed, e.g., the team is no longer winning and thus the defence role is no longer the most important. Similar considerations hold for the *System Objective* that can be different for different game strategic situations. Moreover, since we are dealing with hardware devices, robots may have wrong estimation of their *Self Evaluation* that can be refined over time;

also, robots might fail unexpectedly and messages communicated among robots might be lost, leading the team to have temporarily misaligned knowledge about the current situation.

3. Cooperative Control Method

Hereafter we shall present a cooperative control method to address the problem outlined in sect. 2. The basic idea is to evaluate and share the *Self Evaluation* for the roles that the robots can perform. This evaluation compacts sensory information that each robot collect from the environment in a single value that is shared with the teammates.

Based on the evaluation of the teammates each robot computes the best allocation of robot coalitions to tasks and then decides which role it should execute. Such process is iterated over time to react to environment dynamism, changes in task priorities and possible robot failures or malfunctioning. In particular, the algorithm is run at a predefined execution rate, which is specified according to the application domain, and at each execution all information required to run the algorithm are acquired by the robots through sensor perception or through communication. Therefore, at each execution the algorithm considers the most recently available information that reflect possible changes in the scenario.

The assumptions underlying this method are the following: i) robots are able to compute their *Self Evaluation* for each role depending on their current state and the state of the environment; This is done every time the algorithm is run, by using the information acquired through sensors (see Section 4 for details). ii) each robot can estimate the *Self Evaluation* for each role and each of their team mates. In particular, in our approach the estimates are broadcast to all other teammates. iii) The *System Objective* (i.e., the desired achievement level for each role l_i) is known to all the robots. iv) Tasks to be allocated have priorities which are known to the whole team; task t_i has a higher priority than task t_{i+1}.

Given the above assumptions, the cooperative control method includes three main steps: i) each robot computes the *Self Evaluation* value for each task; ii) robots broadcast the computed *Self evaluation* value, for each task, to all team members; iii) each robot computes the coalitions to allocate to each task based on the information received by team mates. The last step uses a greedy algorithm based on task priority. The three steps are executed continuously over time, to take into account possible unexpected changes in the environment the team need to react to.

Figure 1 reports the pseudo-code description of the allocation method which is executed by every agent. It takes as input the tasks to be executed, the available agents (including the one executing the algorithm), and the desired achievement level for each task; the output is the task which should be executed by the agent running the algorithm. Basically, it sorts the tasks according to their priority (line 1), and then for each task computes the best agent coalition for that task.

To compute the best coalition, the algorithm sorts agents according to their ability to fulfil the task (line 4), and then incrementally builds a set of assigned agents for the task. At each iteration the algorithm checks whether the achievement level of the task has been reached (line 7), by computing the current *Agent Satisfaction* which expresses the coalition value of the current set of allocated agents for that task. The **Aggregate**(*AgentCoalition*) function aggregates the achievement values of the agents and computes the achievement level of the coalition.

The inner loop terminates either because the algorithm found a set of agents that satisfies the achievement level of the task, or because the task is not achievable with the current available agents. In the latter case the algorithm sets the set of assigned agents to be empty (line 13), otherwise it removes from the set of available agents the assigned agents (line 16). Before proceeding for the second task the algorithm checks whether the agent executing the algorithm (*mySelf*) is assigned to the current task; in this case it terminates returning the task to execute. Otherwise the algorithm proceeds to the following task. If the agent is never assigned a special value *NoTask* is returned (line 23).

Notice that the algorithm always terminates because at each iteration of the inner while loop one agent is removed from *SortedAgents* list and in the while condition at line 7 the algorithm checks whether the *SortedAgents* list is empty. Therefore, at most the algorithm repeats the statements inside the while loop until all agents have been included in the coalition. Similar reasoning holds for the outer while loop over tasks.

Also, since allocated agents are removed from the list of available agents (line 16) agents allocated to one task will never be considered for another task, and thus we will never have an agent being part of two different coalitions. Finally, all agents have the

```
1: Input: $Tasks$, $Agents$, $SystemObjectives$, $SelfEvaluations$
2: Output: $TaskToExecute$
3: $SortedTaks \leftarrow$ Sort $Tasks$ given priority
4: while $SortedTasks \neq \emptyset$ do
5:     $Task \leftarrow \mathbf{Pop}(SortedTasks)$
6:     $SortedAgents \leftarrow$ Sort $Agents$ given $Self\ Evaluation$ for $Task$
7:     AgentSatisfaction $\leftarrow 0$
8:     AssignedAgents(Task) $\leftarrow \emptyset$
9:     while $AgentSatisfaction < SystemObjectives(Task) \wedge SortedAgents \neq \emptyset$ do
10:        AssignedAgents(Task) $\leftarrow$ AssignedAgents(Task) $\cup$ $\mathbf{Pop}(SortedAgents)$
11:        AgentSatisfaction $\leftarrow \mathbf{Aggregate}(AssignedAgents(Task))$
12:    end while
13:    if $AgentSatisfaction < SystemObjectives(Task)$ then
14:        $AssignedAgents(Task) \leftarrow \emptyset$
15:    else
16:        $Agents \leftarrow Agents \setminus AssignedAgents(Task)$
17:    end if
18:    if $mySelf \in AssignedAgents(Task)$ then
19:        $TaskToExecute \leftarrow Task$
20:        return $TaskToExecute$
21:    end if
22: end while
23: return $NoTask$
```

Fig. 1. Allocation procedure

same input data, because the set of tasks and the desired achievement levels are known a-priori, and agents communicate their *Self Evaluation* for each tasks. Since all agents execute the same algorithm on the same input data, the allocation to tasks will converge to a common solution.

As for solution quality, let us consider the following property of the domain: Given two coalitions and a task t_j, if the sum of the estimated level of performance for the agents in coalition C' is greater than the sum of the estimated level of performance for the agents in coalition C'' then the value of allocating coalition C' to t_j is higher than allocating coalition C''. Considering the formalism described in Section 2 this amounts to

$$\forall t_j \in T,\ \forall C', C'' \in \mathcal{C} \text{ if } \sum_{i \in R(C')} s_i^j \geq \sum_{k \in R(C'')} s_k^j$$
$$\text{then } F(C', t_j) \geq F(C'', t_j)$$

Where $R(C)$ indicates the set of indexes of the agents in coalition C. When the above property is verified the allocation computed by the agents is optimal with respect to equation 1. Notice that this assumption is verified for example when $F(C, t_j)$ is the sum of the *Self Evaluation* of the agents forming coalition C to perform task t_j.

This is because tasks are sorted according to their priority, and it is always preferable to accomplish a higher priority task than any other lower priority task combination (as stated in Section 2). In addition, for each task, agents to be inserted in the coalition are sorted according to their ability to perform the task. Therefore, if the property above holds, by executing the Algorithm 1 we always evaluate the best coalition for the most important task first, thus maximising 1.

4. Sensor-driving activation of a collective behaviour

In this section we describe how relevant information for the coordination process can be extracted from the robot's sensor perceptions. In particular, we need to aggregate the reading that the robots acquire at each time step to recognise environmental patterns which are relevant for the coordination process and specifically to compute the above mentioned *Self Evaluation* measure for each agent.

The computation of the *Self Evaluation* measure is a key component to ensure a correct behaviour of the overall coordination method. In particular, the data processing method described here is respon-

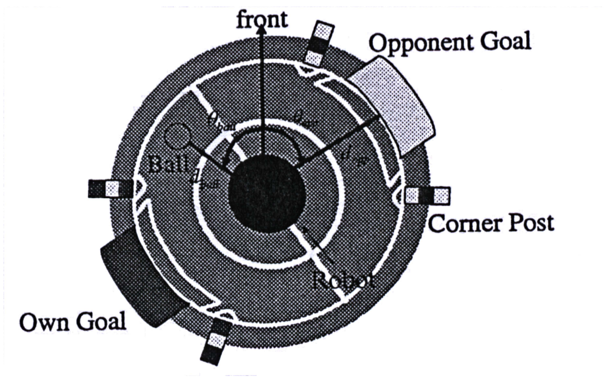


Fig. 2. Omnidirectional vision system

sible to filter out the inherently noisy reading, obtained from the sensors, and provides to the cooperative control method, stable and accurate estimates for the *Self Evaluation*. In the following we detail how this process is realised using an omnidirectional vision system.

4.1. *Vision Depth*

We represent the robot team as a set of *moving points* $\{R_1, R_2, ..\}$ on a plane surface. Now, let us define the objects each robot perceives in the operating field considering that the main sensor system is based on an omnidirectional camera where each object appears reflected on a conic surface with an angle θ, referred to the ahead direction as appears in fig 2. So, θ varies on a 2π range, namely, $-\pi <= \theta <= \pi$ and, the object is positioned at a distance r from the origin of the frame of reference centered on the robot itself.

We introduce on the arena a fixed frame of reference and we consider the positions $\langle x_i, y_i \rangle$ and $\langle x_j, y_j \rangle$ of the robots R_i and R_j, respectively. The distance d between the points can be easily computed by the means of the well-known euclidean formula

$$\begin{aligned} d^2 &= (x_i - x_j)^2 + (y_i - y_j)^2 \\ &= r_i^2 + r_j^2 - 2r_i r_j \cos(\varphi_i - \varphi_j) \\ &= (r_i - r_j)^2 + 2r_i r_j (1 - \cos(\varphi_i - \varphi_j)) \\ &= (r_i - r_j)^2 + r_i r_j (\varphi_i - \varphi_j)^2 \end{aligned}$$

where we have used both Cartesian and polar coordinates, referring to the standard translation formula, and the first term approximation of the cosine in the last equality [2]. Different robot positions result in different evaluations of the term $r_i r_j$; however, we could substitute this quantity for its mean value p^2,

$$p^2 = \frac{2}{N(N-1)} \sum_{i \neq j} r_i r_j$$

over N robot detections in the scene. In this way, the scalar quantity w, defined by the relation

$$w = \sqrt{r^2 + p^2\theta^2} \qquad (2)$$

can be taken as an approximated estimation of the distance d between robots, where $r = r_i - r_j$ is the distance evaluation when the position vectors $\mathbf{r}_i$ and $\mathbf{r}_j$ belongs to the same line [3] and $\theta = \varphi_i - \varphi_j$ is the relative angle between the two robots.

Now, let us generalise the previously discussed scenario by introducing a mobile frame of reference for each robot and let us consider its vision field. If the robot perceives the objects Q_1 and Q_2, they are univocally determined by their position vectors $\mathbf{r}_1$ and $\mathbf{r}_2$, respectively. Again, we compute the distance $r = r_1 - r_2$ as the objects and the robot were on the same straight line and the angle $\theta = \varphi_1 - \varphi_2$ takes into account their relative position in the vision field. We can justify the previous approximation by considering the special case of several objects disseminated around the robot at the same distance from it.

Because they are visible under different angles but they belong to the same circle, their displacement can be evaluated along the circle: their relative position doesn't change by overestimating their relative distance. Nevertheless, in the general case, the objects around the robot are positioned at different distances: we choose a reference circle with the respect to which we evaluate their relative distance. To this aim we consider a circular strip, where the objects of interest are positioned and whose middle circumference is taken as a reference circle by linking it to the conic surface of the vision system. Then, we project all objects on it but, firstly, we record their relative distance from it; the term r^2 is merely used to correct the approximated distance evaluation on the reference surface [4] by explicitly considering they are positioned on different conic surfaces [5].

[2] We shall justify this approximation in the next paragraph.
[3] Namely, we evaluate the distance between the two circumferences with radii r_i and r_j, respectively, and centered on the fixed frame of reference.
[4] The middle circumference on the circular strip, in our model.
[5] Circles in our model.

Considering this, the parameter p in eq. 2 denotes the *vision depth* of the omnidirectional vision system so that an object appears reflected as a reference point, with $p\theta$ a linear coordinate expressing the distance on the conic surface[6] referred to a well-specified point ahead the robot. Moreover, if r_{min} and r_{max} are the radii of the circles[7] including all the observed objects, then

$$p = \frac{r_{min} + r_{max}}{2}$$

so that we could assign the vector $\mathbf{w} = \langle r, p\theta \rangle$ to every visible object falling in the robot vision field. If two or more objects fall in the vision field we have $\langle r_1, p\theta_1, r_2, p\theta_2, r_3, p\theta_3, ...\rangle$ yielding to a more general coordinate system.

4.2. *Focusing Lens*

The vision depth previously discussed is useful to focalise the objects of interest for the robot task. Changing the value of the parameter p the robot focuses on different environment configurations as they were different layers of a complex scene. A more subtle control of the scene could be implemented if each layer were accessible with a different focusing property. To this aim we shall introduce a transformation matrix to modify how vision parameters are acquired. The most general transformation $H(\varphi)$, which doesn't warp the relative angles of an object after a rotation of an angle φ around a vertical axis, takes the form

[6] Within our model a circle positioned at the distance p from the robot.
[7] The circular strip, in the preceding discussion.

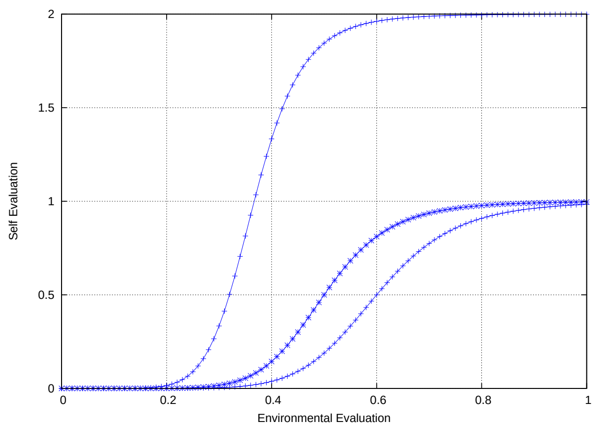


Fig. 3. Self Evaluation Example

$$H(\varphi) = p \begin{bmatrix} \frac{1}{qC_2}\cos\varphi & -\frac{1}{pC_1}\sin\varphi \\ \frac{1}{qC_2}\sin\varphi & \frac{1}{pC_1}\cos\varphi \end{bmatrix} \tag{3}$$

whose effect is to map the vector $\mathbf{w}$ into the new vector $\mathbf{w}^{'}$ having the magnitude

$$w^{'} = p\sqrt{\left(\frac{r}{qC_2}\right)^2 + \left(\frac{\theta}{C_1}\right)^2} \tag{4}$$

where p is the vision depth, q is a given constant which takes into accounts the distance unit, whereas C_1 and C_2 are two dimensionless arbitrary quantities which work as scaling factors. If we choose the preceding parameters to satisfy the following identity

$$pC_1 = qC_2 \tag{5}$$

the transformation matrix given by eq. 3 simplifies to

$$H(p) = \frac{1}{\mathsf{C}_1} \begin{bmatrix} \cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi \end{bmatrix} \tag{6}$$

Under this particular condition, the distance w from an object in the scene, defined in this frame of reference, yields to

$$w^{'} = \frac{w}{pC_1} \tag{7}$$

and it has the following interpretation: *a robot while focusing on an event in the environment, situated around a distance p, can magnify it by a factor C_1.*

4.3. *Tracking an Object*

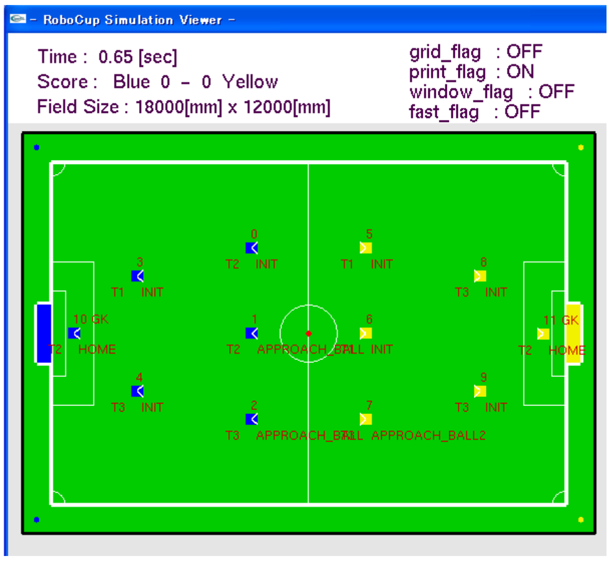
The task completion of an individual teammate could require a behaviour to properly track an object Q, initially positioned in the point represented by the vector $\mathbf{w}$ according to the omnidirectional vision system of the robot. In many cases only the distance w from the mobile frame of reference is actually necessary but the vision system could magnify it by properly scaling the object. The required transformation is made by the matrix $H(\varphi)$ applied to $\mathbf{w}$. In our case, it reduces to eq. 7 which can be taken as a basis to estimate the relevance of the object for the task completion, by considering its negative exponential weight

$$E(w) = \lambda \exp(-\frac{w}{\mathsf{k}p}) \tag{8}$$

where p is the *vision depth* parameter and k represents the *magnify factor*, through which the individual robot is monitoring the object Q. In fact, eq. 8 could be assimilated to a component term of a *partition function* in the almost same fashion it appears in statistical mechanics. We shall use this function as a basis to evaluate the progress of the collective task. In (7) Fujii has shown actual computations of the quantity $E(w)$ within different scenarios. Finally, notice that λ plays the role of *normalisation factor*, which could be evaluated by either summing up all the terms like eq. 8, where w_i is the variable parameter, or by integrating the same terms on the proper ranges of variability to include all the points of interest. However, in the applications such constant values are established experimentally in order to adjust possible sensor distorsions. If we choose the normalisation over the range 0..1, then we can interpret $E(w)$ as a *probability density function*, which resembles the Boltzmann distribution in statistical mechanics with the temperature substituted for the vision depth, and k_j dimensionless constant quantities.

Some complex situations require two or more objects to be tracked at the same time; in such cases different solutions can be devised, for example, by squaring the relevant components and also taking into account the possible *alignment degree* between objects

$$E(w_1, w_2) = \lambda e^{-\sqrt{\left(\frac{w_1}{\mathsf{k}_1 p_1}\right)^2 + \left(\frac{w_2}{\mathsf{k}_2 p_2}\right)^2 + \left(\frac{\theta_1 - \theta_2}{\mathsf{k}_{12}}\right)^2}} \qquad (9)$$


- RoboCup Simulation Viewer -

Time : 0.65 [sec]
Score : Blue 0 - 0 Yellow
Field Size : 18000[mm] x 12000[mm]

grid_flag : OFF
print_flag : ON
window_flag : OFF
fast_flag : OFF

Fig. 4. Simulation environment

The rationale of this approach is the interpretation of the objects detected by each individual robot. This is done by comparing the observed positions of specific reference objects with their expected positions. Specifically, the robots evaluate the *most probable distribution* of the objects with respect to relevant patterns to be recognised. The continuous monitoring of such patterns, through probability distribution evaluation, drives the computation of the *Self Evaluation*. The specific computation of the *Self Evaluation* for the RoboCup scenario is detailed in the next Section.

5. RoboCup Scenario

In this section we specify the computation of the cooperative control method described in Section 3 and the extraction process for the required information described in Section 4 to the RoboCup scenario. In particular this amounts to specify how the *Agent Satisfaction* and the *Self Evaluation* are computed. We then present the simulated environment used to evaluate our approach and the obtained results.

5.1. *Self Evaluation*

Let us consider the coalition formation problem from the point of view of each individual robot, whose movement depends on the trajectories of all its teammates. In such a general case, eq. 7 generalises to

$$E(w_1, w_2, \ldots) = \exp\left(-\sqrt{\sum_{i=1}^{N}\left(\frac{w_i}{\mathbf{k}_{\mathbf{i}} p}\right)^2}\right) \qquad (10)$$

where N is the number of objects [8] to be tracked and we assume that every object is differently focused by the vision system but no alignment among objects is taken into consideration.

However, the more objects are counted the more computation is needed to evaluate the actual probability distribution against the expected one; so, the implemented coordination schema considers either one or two objects to be focused at the same time. The **environmental evaluation** E of the event in the scene is given through eq. 8 or eq. 9 and the resulting value can be understood in terms of prob-

[8] For example, number of teammates, opponents, ball when we explicitly consider the RoboCup scenario.

ability. Nevertheless, a more useful quantity is obtained by means of the following definition

$$S(w) = \frac{\alpha E(w)}{1 + E(w)} = \frac{\alpha}{1 + \lambda \exp(\frac{w}{\mathsf{k}p})} \tag{11}$$

so that its rate of change due to the object moving within the vision field, takes the form of the generalised logistic function

$$S^{'}(w) = -\frac{S(w)}{\mathsf{k}p}(1 - \frac{S(w)}{\alpha}) \tag{12}$$

and it is the basis for the **quantitative self-evaluation** of the *offence*, *support* and *defence* tasks. In the preceding equation $\mathsf{k}p$ defines the inverse growth rate whereas α is the asymptotic value of S(w).

Three compatible instantiations of the sigmoidal function for the mentioned features are reported in fig. 3. They are taken as a basis for the application examples discussed below in the section where, instead of using directly the sigmoid function, we have used the straight-line approximations, in the same fashion they appear in (7) and they are specified as follows:

– **offence**, which refers to the upper line in fig. 3,

$$S_{off} = \begin{cases} 2 \ if \ E_{off} > 0.5 \\ 1 \ if \ 0.2 < E_{off} \leq 0.5 \\ 0 \ if \ E_{off} \leq 0.2 \end{cases} \tag{13}$$

– **support**, which refers to the lower line in fig. 3,

$$S_{supp} = \begin{cases} 1 \ if \ E_{supp} > 0.6 \\ 0 \ if \ E_{supp} \leq 0.6 \end{cases} \tag{14}$$

– **defence**, which refers to the middle line in fig. 3,

$$S_{def} = \begin{cases} 1 \ if \ E_{def} > 0.5 \\ 0 \ if \ E_{def} \leq 0.5 \end{cases} \tag{15}$$

The experimental evidence has suggested us the choice of the specified thresholds. The thresholds are also experimentally tuned to avoid passive oscillations during sensor data acquisitions. More details on possible implementations are discussed by Fujii in (7).

5.2. *Agent Satisfaction*

The *Agent Satisfaction*, discussed in Section 2, drives the task allocation process as explained in Section 3. However, when applied to the RoboCup Soccer scenario, specific domain knowledge are used to make the algorithm more effective and efficient.

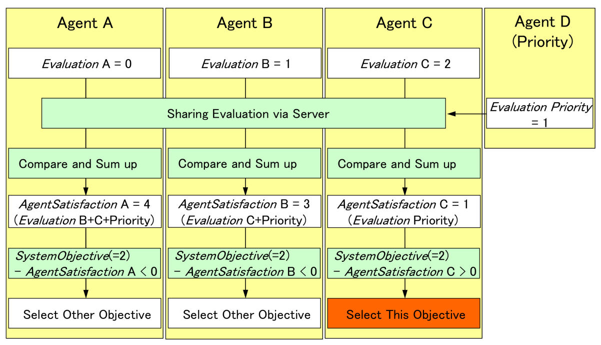

Agent A
Agent B
Agent C
Agent D (Priority)
Evaluation A = 0
Evaluation B = 1
Evaluation C = 2
Evaluation Priority = 1
Sharing Evaluation via Server
Compare and Sum up
Compare and Sum up
Compare and Sum up
AgentSatisfaction A = 4 (Evaluation B+C+Priority)
AgentSatisfaction B = 3 (Evaluation C+Priority)
AgentSatisfaction C = 1 (Evaluation Priority)
SystemObjective(=2) − AgentSatisfaction A < 0
SystemObjective(=2) − AgentSatisfaction B < 0
SystemObjective(=2) − AgentSatisfaction C > 0
Select Other Objective
Select Other Objective
Select This Objective

Fig. 5. Effective cooperation among robots

For example, in the context of RoboCup soccer middle size league, robots belonging to the same team are usually heterogeneous. In particular, most of the teams have a specialised robot to act as the goalkeeper. For this role the designed robot is always desirable to any other robot even if the *Self Evaluation* value for the role *goal keeping* of the other robots might be higher.

To take this aspect into account, we introduce the concept of *priority* of a robot over the other robots for each task. Conceptually we want that if a robotic agent R_i is somewhat specialised for a task t_j, then R_i will always be considered, before all the not-specialised robotic agents, in the computation of the *Agent Satisfaction* for task t_j. To this aim we actually order the list of agents for a given task, based on the agents' *priority* first and then on the self evaluation for that task. This results in changing line 6 of Algorithm 1 to sort agents reflecting their *priority*. Notice that priority differs from the *Self evaluation* of a robotic agent for a task. The former represents a static concept that does not change over a specific game, i.e., a goalkeeper may have specialised hardware that will help him to defend the goal better than any other robotic agents in the team. On the other hand, the *Self Evaluation* depends on dynamic properties of the robotic agents, which change very rapidly during the game (e.g., position and orientation). By sorting the robotic agents according to priority first we ensure that an agent that is specialised for a task will always be preferred over a non specialised one. However, by sorting robotic agents that have the same priority, according to their *Self Evaluation*, we ensure that among robotic agents having the same level of specialisation for a task, we choose the most capable ones.

Moreover, in the RoboCup scenario the aggregation function used in 1 to compute the *Agent Satisfaction* is the sum of each agent's *Self Evaluation* value.

Therefore, summarising the cooperative control method proceeds as follows:

Step 1. Each agent computes its own *Self Evaluation* for all tasks, according to the current perceived situation.

Step 2. Each agent broadcasts the values of *Self Evaluation* for all tasks, to all its team mates.

Step 3. Each agent executes Algorithm 1, the algorithm computes the *Agent Satisfaction* considering the most recent available information (e.g., most recent values for *Self Evaluation* communicated by team mates). The algorithm returns the task that agent will execute, according to the computed *Agent Satisfaction* and the predefined *System Objective*.

Figure 5 shows a graphical scheme of a specific execution of the cooperative method described above for a specific task. As one can see, at the end of the algorithm the coalition for the task includes agent C and D. Agent D belongs to the coalition because it is the only one having priority for the task, while agent C belongs to the coalition because it has the highest *Self Evaluation* among the remaining agents.

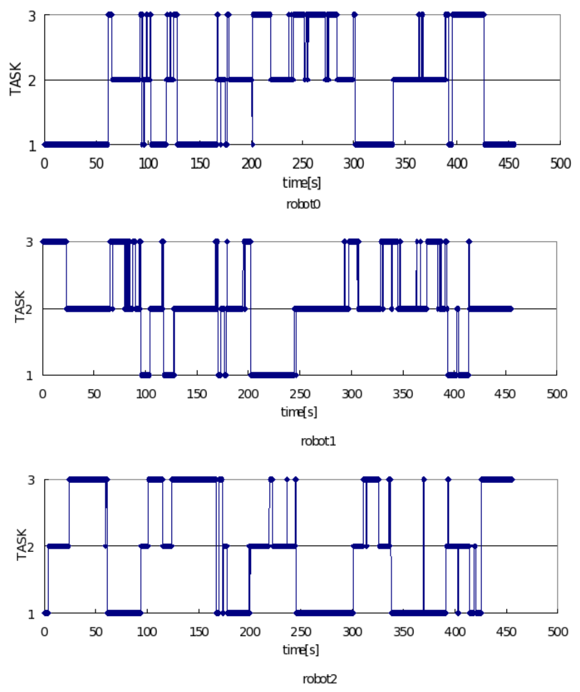


Fig. 6. Role exchange among 3 robots

Agent A and B do not belong to the coalition and will thus focus on other tasks, because they know that the *System Objective* will be reached by the selected coalition and therefore their contribution is not required.

From a different point of view (12), the execution of Algorithm 1 can be interpreted as a social rule; where each robot is required to evaluate the achievement level of all other robots by comparing their behaviour with its own task achievement. This is a kind of *individual-social evaluation* and it motivates why we use the term *Agent Satisfaction*.

As a final remark, we can observe that the robustness of the method strongly stems from the simplicity of the algorithm so that its execution is taken at very high rate; data are exchanged very often and thus, misaligned knowledge appears only for short time interval. Moreover, since the allocation is computed very frequently, the algorithm computes the allocation with freshly updated values for the *Self Evaluation*. This ensures good quality for the allocation even when the system faces abrupt changes of the system configuration, such as robot failures

5.3. *Simulation Environment*

In the Robocup *Middle Size League* scenario, the general situation described in Sects. 2, 3 and 4 simplifies as follow. The multiagent team is made up on three kinds of robot: the *Goal-keeper* KP, the *Defender* DF, both using a differential drive, and the *Field Player* FP, equipped with an omnidirectional drive. Because we have four such kinds of agent we can write

$$\begin{aligned} R_1 &\in GK, \\ R_2 &\in DF, \\ R_3, R_4, R_5, R_6 &\in FP \end{aligned} \tag{16}$$

However, the roles (tasks) we want to assign them are *Defense*, *Support* and *Offense* and each agent estimates its own ability [9] to perform one of the previously listed tasks accordingly to the formulas 13, 14 and 15. Thus, the agent's capabilities S_i [10] to perform each task istantiates to

$$\begin{aligned} S_i &= \{s_i^1, s_i^2, s_i^3\} \\ &= \{S_{def}, S_{supp}, S_{off}\} \end{aligned} \tag{17}$$

[9] *Self Evaluation*
[10] which is a vector of real values

Now, in this specific situation, we can optimise the computation of the agent satisfaction performed in the inner loop of the allocation procedure 1. Specifically, we can avoid sorting the agents according to priorities and then according to their self evaluation by computing directly the agent satisfaction. We can do this by using the following formulas:

$$V_k(l) = \sum_{i=1}^{p} s_i^k + \sum_{i=p+1}^{n} g_i^k(l)$$
$$g_i^k(l) = \begin{cases} s_i^k \ if \ s_i^k > s_l^k \\ 0 \ \ if \ s_i^k \leq s_l^k \end{cases} \tag{18}$$

where p is the number of the prioritized agents and $V_k(l)$ is the agent satisfaction referred to the task t_k computed from the point of view of the agent R_l. The computation proceeds by considering that agents R_1 and R_2 are prioritized agents, while the self evaluation of the others are compared with the self evaluation of the agent which is executing the computation (recall that this computation is made concurrently by each agent involved in the coordination process).

This cooperation schema, devised accordingly to the previous sections, is central in controlling the collective behaviours of the team, by the activation and maintenance of the required individual behaviours.

To the aim of a more precise behaviour analysis, we have implemented the cited algorithm in a simulated RoboCup environment. Before discussing the obtained results, we briefly describe the simulation environment we used. In fig. 4 it is shown the 2D simulated field where two teams of six individual robots are disposed at the beginning of the simulation. The ball is positioned, as in the competition, at the center of the field and data are collected during a single test as reported below. The opponent team is just a clone of the evaluating team so that the coordinating algorithm is equally executed on both teams.

The simulated vision system has been properly devised to extract all the relevant information to estimate the *Self Evaluation* for every agent R_i. This is computed according to the discussion of Section 4 where each robot must recognise some critical objects, given the depth of vision field and the related magnification. As it has been already pointed out, some constants have been determined empirically and also the thresholds which definitely assign integer values to those quantities.

Each agent knows the *System Objective* (the desired level of achievement on the completion of the task).

5.4. *Results obtained in the simulated environment*

The results reported here are for a scenario involving three tasks: $\langle Defence, Offence, Support \rangle$. The *Offence* task represents the behaviour of *carrying the ball towards the goal*. A robotic agent is in the best position to perform this task when it is near the ball and when the robot position, the ball position and the center of the goal are aligned. The evaluation of the *Support* objective is computed based on the distance between the actual robot position and the position needed to support other robots. On the contrary, the *Defence* objective evaluation only de-

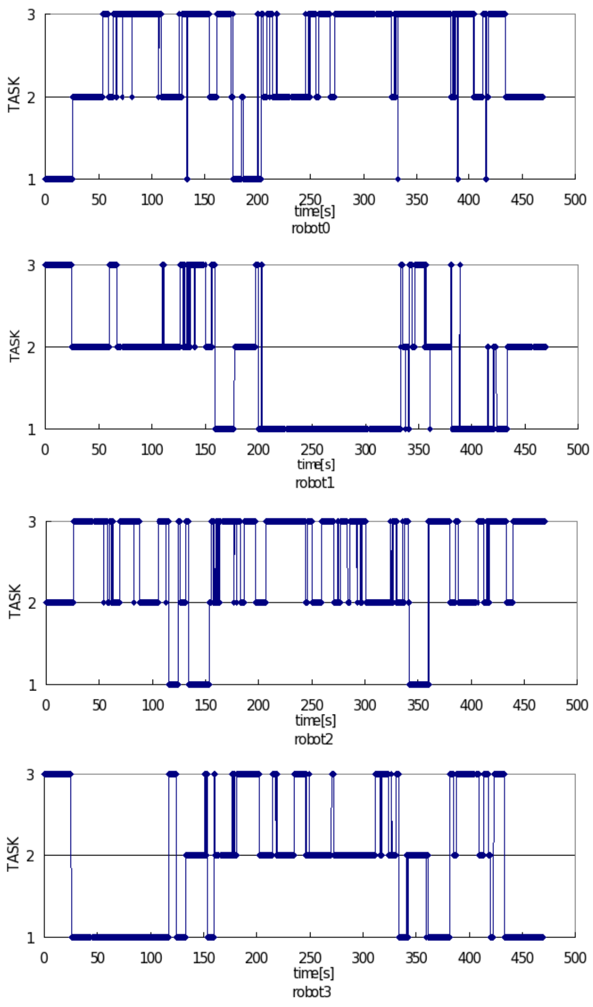


Fig. 7. Role exchange among 4 robots

pends on the position of the robot. Robots have a high defence evaluation value when they are placed between the ball and the goal. The computed values of the *Self Evaluation* are obtained according to the eqs. 13, 14 and 15.

Figure 6 reports results for three agents cooperating while Figure 7 reports results for four agents. In both figures we report the task that each robot is performing over time. Consequently, on the x axis we report time while on the y axis we report a numerical code for the tasks: 1 is Defence, 2 is Offence and 3 is Support.

Figure 6 shows how the coordination algorithm is able to balance the effort of the three robots on different tasks. In particular, while the Defence task is almost always carried out by only one robot, both the Offence and Support tasks involve more robots forming coalition and collaborating on the same task. This is because these tasks benefit from the cooperation of more robots, and therefore the coordination algorithm will try to allocate more team mates to these tasks.

In Figure 7, it is possible to see that a similar behaviour can be observed when four robots are cooperating. In this case the amount of time that more than one robots are executing the same task is obviously higher, but as before, the Defence task is almost always carried out by a single robot, while robots cooperate more on the Offence and Support tasks. As before the coordination algorithm is also able to balance the team effort in order to cover all the roles and thus having an effective coordination.

6. The RoboCup Middle Size League testbed

In this section we briefly present the control architecture used by the EIGEN team in the RoboCup middle size soccer competitions. Moreover, we discuss the results obtained by the application of the

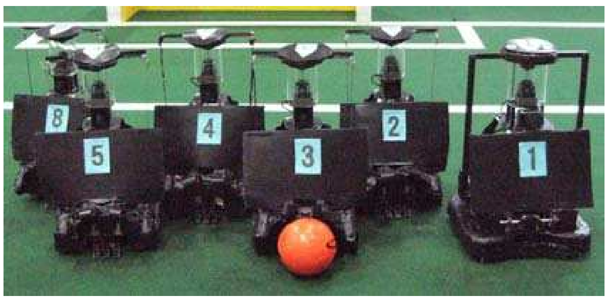
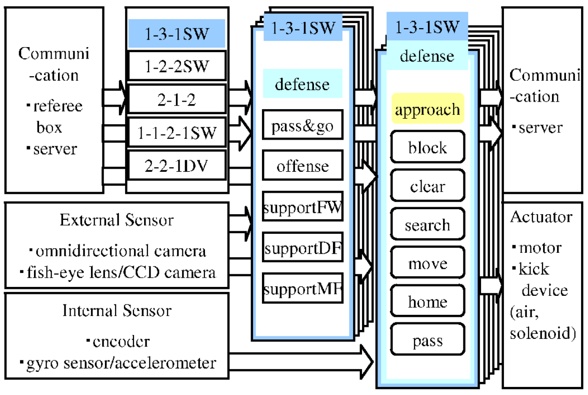


Fig. 8. EIGEN Team



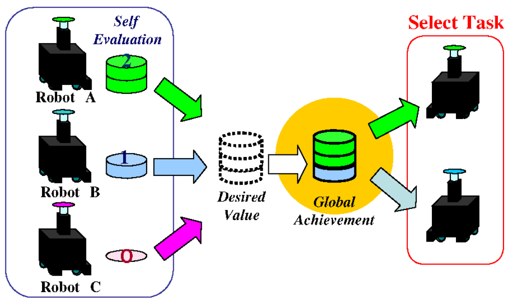


Fig. 9. The flow of information during cooperation

proposed cooperative control method in the competitions.

6.1. *Cooperative Control in the RoboCup scenario*

The EIGEN team robots are shown in fig. 8. Each teammate has an omnidirectional drive system with four roller wheels. The motivation is that its employment improves the movement and the stability of the platform also by increasing its capability to change direction. Each robot is equipped with an omnidirectional vision system. This system has the same features as it has been presented at RoboCup2005, described in (7) and (6). However, since the year 2007 some robots have been equipped with two extra cameras and a gyro sensor in addition to the omnidirectional vision system.

The control architecture is shown in fig. 11; environment information, acquired by each robot through its own sensors, are used mainly to recognise white lines and landmarks. Hence, it calculates its position and orientation, which are communicated to the other robots, allowing it to share information about the ball, relative teammates positions, task achievement evaluation and so on.

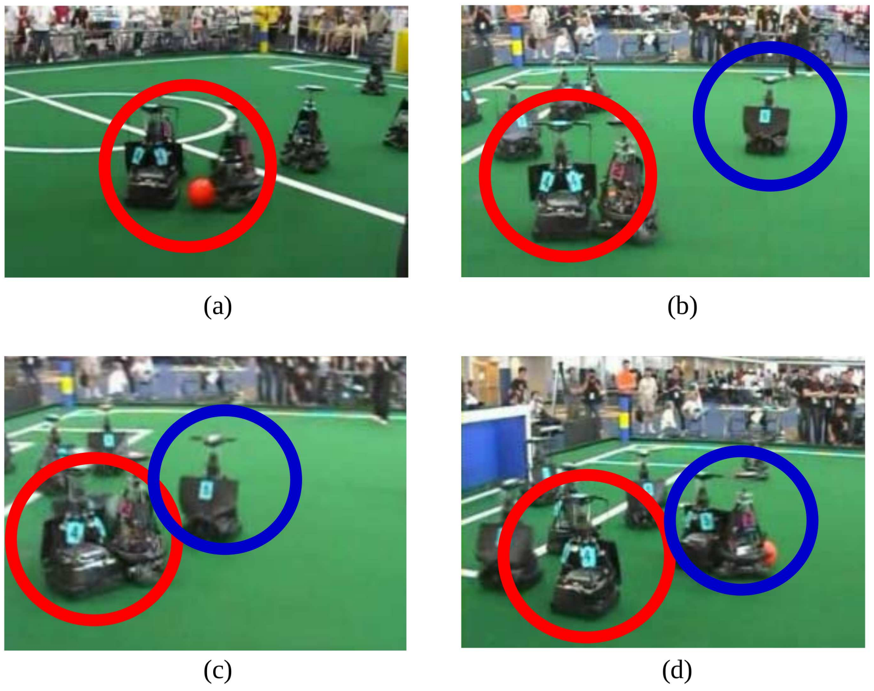


Fig. 10. Cooperative action at RoboCup 2007

Evaluating the task achievement of the group results in both selecting a group formation and the task assignment for each robot, firing the appropriate action module. In particular, the formation module provides the task assignment algorithm with the priority on the tasks. These modules are designed as *action schema* to solve a well-specified task, such as Offence, Support and so on. Finally the robot generates the action by the *extended Fuzzy Potential Method* (19) according to the environment information.

The cooperative method is the one described in Section 5.2 but now the environment information are directly acquired from sensors, and the information required by the cooperative control method are computed as described in Section 5.1. In particular, robots do not use any explicit synchronisation, they share information at each iteration of the algorithm, using an UDP protocol, and they use the last received message from each team mate as the current valid information. To avoid using out of date information, if messages from a team member are not received for a predefined period of time then the information regarding that team mate are ignored.

Figure 9 reports a detailed view of the information flow for the multirobot system used in the actual competitions.

6.2. *Experimental Results*

As previously mentioned the coordination method described here was used by the EIGEN team within the RoboCup competitions since 2004. The method was able to effectively coordinate the robotic platform during the competition, and it constitutes one of the key reasons for the excellent performance of the team in the RoboCup competitions.

To provide a sample of how the coordination mechanism behaved during the actual competition we report in Figure 10 a sequence of images from RoboCup 2007 (11). In (a) and (b), the marked robot in the lower left corner tried to get ball from its opponent and the marked robot in the upper right corner was supporting it. In (c), the lower left marked robot could not get the ball, because of prevention by opponents. In (d), the upper right marked robot took the offence role and eventually got the ball. Similar scenes were often shown during the competitions.

Videos from the RoboCup competitions showing relevant coordination actions can be seen at `http://www.yt.sd.keio.ac.jp/robocup/eigen_movie.html`.

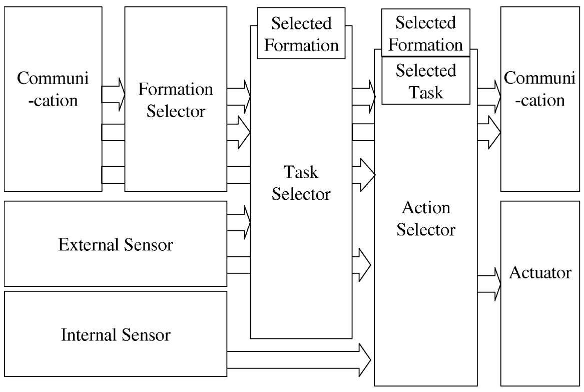


Fig. 11. Cooperation architecture

7. Related Work

Cooperation in multirobot systems has been addressed with several different approaches as discussed in (5). Cooperation approaches range from not coordinated, to weakly coordinated (where each robot considers only simple state information about team mates when choosing its actions) and strongly coordinated approaches (where robots coordinate their action using a specified coordination protocol). Strongly coordinated approaches are frequently used when dealing with complex scenarios as the one we consider in this paper.

Within the strongly coordinated approaches, ALLIANCE (14) and BLE (21) are examples of behaviour-based approaches to multi-robot task allocation. In the former motivational behaviours monitor and dynamically reallocate tasks by providing fault tolerance and adaptiveness. In the latter, each robot executes a task through a specific behaviour. Task selection is implemented by continuously broadcasting locally computed utilities using a greedy algorithm to determine the most useful task. Another behaviour-based example of task allocation in multi-robot systems stems from the concept of *vacancy chains* as discussed by Dahl and Mataric (3). This approach is implemented in groups of homogeneous robots where vacancy chains emerge through reinforcement learning.

In contrast to these works, our work is based on the higher level concept of roles and tasks rather than behaviours. Moreover in our work robots explicitly form coalitions to execute tasks.

The task assignment problem has been frequently addressed in multirobot systems using market-based methods (1; 8; 4; 22) In market-based approaches robots bid and negotiate to obtain tasks. The negotiation process can be of various types but auctions are often used. For example, in TraderBots (4), an auction mechanism, through a revenue/cost mapping function, greedily assign tasks to the highest bidders In this system, a RoboTrader module on each robot coordinates the activities of the agent and its interactions with other agents. Specialised dynamic role assignment methods have been used for robotic soccer, as in Pagello et al. (13) and Stone and Veloso (18), where the robots dynamically switch between roles such as attacker and defender or master and supporter. Burgard et al. (2) consider task allocation and coordination for multi-robot exploration. For each robot, the trade-off between the utility and cost of potential target points are evaluated for exploration with the aim to be properly assigned to each robot.

In contrast to these works in this paper, we focused on a coalition formation problem rather than task assignment.

Coalition formation has been also addressed in multirobot systems. Parker and Tang (15) address the problem of single-task robots performing multi-robot tasks while developing heterogeneous robot coalitions that solve single multi-robot tasks. ASyMTRe (Automated Synthesis of Multirobot Task solutions through software Reconfiguration) is the paradigm they use to generate multi-robot coalitions using complete information and it has been implemented on tasks that require multiple robots to share sensor and effector capabilities. Vig and Adams' approach (20) refers to a multi-robot coalition formation algorithm which uses an adaptation of Shehory and Kraus algorithm (17). The algorithm comprises two stages: in the first one robots compute the initial coalition values for all possible coalitions in a distributed way; in the second stage of the algorithm, robots agree on which coalitions should be formed.

In contrast to these works, here we focus on devising a specific coalition formation algorithm able to work in a very dynamic and complex scenario. In particular with respect to (20) we focus on a simpler setting, where tasks have a predefined priority and the value of a coalition can be computed by summing up the *Self Evaluation* [11] of the coalition members;

[11] Recall that the *Self Evaluation* is an estimation of the capability to perform a task, thus summing up the *Self Evaluation* agents are effectively summing up their estimation of the capabilities

in this way we can avoid the computation of all the possible coalition values needed for the algorithm presented in (17) and (20). Avoiding this computation in our setting is important because the value of a coalition is dependent on variables which are very dynamic (e.g., robot's heading, ball position etc.) and subject to uncertainty in the value estimation.

8. Conclusions

In this paper we present an approach to cooperative control based on objective achievement. The approach is explicitly designed for dynamic uncertain environments, and solves a distributed coalition formation problem. This is done by aggregating the information that each agent acquires from the environment and the information communicated from the team mates.

We apply this approach to the RoboCup Soccer domain, and show how the aggregated information required by the algorithm are extracted from the sensor readings of the robotic platforms. When specialised to the RoboCup soccer domain our method is able to optimise the allocation of robot coalitions to tasks in a distributed and efficient way.

The approach has been empirically evaluated both in a simulated environment and during the RoboCup Middle Size League competitions by the EIGEN team. The results obtained show that the approach is able to balance the effort of the robotic platform on different tasks providing an efficient and effective coordination mechanism.

References

[1] S. C. Botelho and R. Alami. M+: a scheme for multi-robot cooperation through negotiated task allocation and achievement. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, pages 1234–1239. Detroit (Michigan), May 1999.

[2] W. Burgard, M. Moors, C. Stachniss, and F. Schneider. Coordinated multi-robot exploration. *IEEE Transactions on Robotics*, 21(3):376–386, June 2005.

[3] T. S. Dahl, M. J. Mataric, and G. S. Sukhatme. Multi-robot task allocation through vacancy chains. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, pages 2293–2298. Taipei (Taiwan), Sep. 2003.

[4] M. B. Dias and A. Stentz. Opportunistic optimization for market-based multirobot control. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 2714–2720. Lausanne (Switzerland), Oct. 2002.

[5] A. Farinelli, L. Iocchi, and D. Nardi. Multirobot systems: a classification focused on coordination. *IEEE Transactions on Systems, Man, and Cybernetics*, 34(5):2015–2028, Oct. 2004.

[6] H. Fujii, M. Kato, and K. Yoshida. Cooperative action control based on evaluating objective achievements. In *RoboCup 2005: Robot Soccer World Cup IX*, number 4020 in Lecture Notes in Computer Science, pages 208–218. Springer, Osaka (Japan), 2005.

[7] H. Fujii, D. Sakai, and K. Yoshida. Cooperative control method using evaluation information on objective achievement. In *Distributed Autonomous Robotic Systems (DARS 04)*, pages 211–220. Springer, Toulouse (F), June 2004.

[8] B. P. Gerkey and M. J. Mataric. Sold! auction methods for multi-robot coordination. *IEEE Transactions on Robotics and Automation*, 18(5):758–768, Oct. 2002.

[9] B. P. Gerkey and M. J. Mataric. A formal analysis and taxonomy of task allocation in multirobot systems. *Int. Journal of Robotics Research*, 23(9):939–954, Sep 2004.

[10] L. Iocchi, D. Nardi, M. Piaggio, and A. Sgorbissa. Distributed coordination in heterogeneous multi-robot systems. *Autonomous Robots*, 15(2):155–168, 2003.

[11] K. Kawakami, S. Makishima, T. Shimizu, K. Morisaki, and K. Yoshida. Eigen team description. In *RoboCup 2007: Robot Soccer World Cup XI Preproc.*. Springer (cd-rom), 2007.

[12] M. Mataric. Issues and approaches in the design of collective autonomous agents. *Robotics and Autonomous Systems*, 16(2-4):321–331, 1995.

[13] E. Pagello, A. D'Angelo, and E. Menegatti. Cooperation issues and distributed sensing for multirobot systems. *Proc. of the IEEE*, 94(7):1370–1383, July 2006.

[14] L. E. Parker. Alliance: an architecture for fault tolerant multirobot cooperation. *IEEE Transactions on Robotics*, 14(2):220–240, April 1998.

[15] L. E. Parker and F. Tang. Building multirobot coalitions through automated task solution synthesis. *Proc. of the IEEE*, 94(7):1289–1305, July 2006.

[16] T Rahwan, S. D. Ramchurn, A. Giovannucci,

V. D. Dang, and N. R. Jennings. Anytime optimal coalition structure generation. In *Proc. of the 22nd conference on artificial intelligence (AAAI-07)*, pages 1184–1190, Vancouver, Canada, 2007.

[17] O. Shehory and S. Kraus. Methods for task allocation via agent coalition formation. *Artificial Intelligence*, 101(1-2):165–200, May 1998.

[18] P. Stone and M. Veloso. Task decomposition, dynamic role assignment, and low-bandwidth communication for real-time strategic teamwork. *Artificial Intelligence*, 110(2):241–273, June 1999.

[19] R. Tsuzaki and K. Yoshida. Motion control based on fuzzy potential method for autonomous mobile robot with omnidirectional vision. *Journal of the Robotics Society of Japan*, 21(6):656–662, 2003.

[20] L. Vig and J. A. Adams. Multi-robot coalition formation. *IEEE Transactions on Robotics*, 22(4):637–649, Aug. 2006.

[21] B. B. Werger and M. J. Mataric. Broadcast of local eligibility for multi-target observation. In *Proc. of the Int. Symposium on Distributed Autonomous Robotic Systems*, pages 347–356. Springer, Knoxville (Tennessee), Oct. 2000.

[22] R. Zlot and A. Stentz. Complex task allocation for multiple robots. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, pages 1515–1522. Barcelona (Spain), 2005.