Local Navigation Strategies for a Team of Robots

Antonio Sgorbissa
Laboratorium
DIST – University of Genova
antonio.sgorbissa@unige.it

Ronald C. Arkin
Mobile Robot Laboratory
College Of Computing - GaTech
arkin@cc.gatech.edu

Abstract--*Whenever a mobile robot has to deal with an environment that is totally or partially unknown or dynamically changing, local navigation strategies are very important for the robot to successfully achieve its goals. Unfortunately, local navigation algorithms that have been proposed in the literature offer poor performance (or even fail) whenever the geometry of the free space in which the robot is requested to operate increases its complexity. In this paper, we deal with a team composed of many robots, and we show how robots navigating within an unknown environment with local communication capabilities (only line-of-sight communication is allowed) can cooperate by helping each other to achieve their own goals.*

Keywords-- *Navigation algorithms, multi-robot exploration, multi-robot cooperation****.***

1 INTRODUCTION

Whenever a mobile robot has to deal with an environment that is totally or partially unknown or dynamically changing, local navigation strategies are very important for the robot to successfully achieve its goals, whether it is requested to accomplish a complex mission or simply to survive in the environment. By local navigation strategies we mean algorithms that return, at each computation step, the best direction for the robot to move on the basis of its current goal and the current perception of its surroundings.

Even if in many cases the robot has an approximate knowledge of the environment in which it is requested to operate and therefore hybrid solutions deliberative/reactive are possible [1][2], local strategies are important for the robot to react in real-time to unforeseen situations; e.g., in the case of Service Robotics applications [3][4], where robots are requested to execute repetitive tasks (e.g., carrying objects, escorting people) within a populated building such as a hospital, or a museum. Moreover, whenever the environment is totally unknown and the robot can rely only on its own perceptions to achieve its goals, the availability of effective, local strategies for exploring the environment becomes even more important in determining the successful accomplishment of a given mission. Consider, for example, high risk military applications where robots are deployed and requested to explore unknown buildings (possibly occupied by hostile robots/individuals) in order to search and report the existence of any hazardous material or to communicate to a remote station other kinds of information which will reduce the risk of losing human lives [5]. In the following, we will take this as our reference scenario.

Unfortunately, local navigation algorithms that have been proposed in literature offer poor performance (or even fail) whenever the geometry of the free space in which the robot is requested to operate increases its complexity: artificial potential field based approaches have the tendency to lead the robot into local minima [6]; search algorithms [7] may require a long time for the robot to find a path to its goal, and are therefore inefficient whenever the time spent in exploring the environment is a factor that needs to be minimized. However, in many applications related to autonomous exploration, the use of a team of robots for maximizing the coverage of a given area is foreseen (as is the case of our high risk reference scenario). Thus, it is straightforward to try to improve the performance of each individual robot (which is given a sequence of navigation tasks and a local navigation algorithm with poor performance) by allowing teammates to cooperate by helping each other to achieve their own goals.

In [8] a formal comparison between the use of explicit and implicit communication is presented, focusing on the improvement of performances of the whole system when more robots are allowed to share with each other their own goals or their own internal state. In our system we allow the use of explicit communication between robots; however we assume that the robots are allowed to communicate only when they are in sight of each other (that is, broadcasting information to all robots throughout the building is not allowed). Limiting the system to line-of-sight communication is a natural consequence of the particular scenario we have chosen; in fact we want the robots to be able to operate in potentially hostile settings where standard radio communication may be impossible. However, as it will be shown in the following, this work does not focus on defining strategies for keeping robots in sight of each other during multi-robot exploration and coverage of the environment. Instead, we consider line-of-sight as a constraint of the problem. This line-of-sight constraint posed for communication seems to fit well with the local/reactive approach to the navigation problem: since the robots rely only on 'local knowledge' of the environment, they are assumed to allow only 'local communication'.

For example, in our reference scenario, we could imagine a situation as depicted in Figure 1: M robots are deployed in an unknown (or partially known) building with the purpose of exploring it and reporting to a base station what they have discovered. A possible strategy could be the following: one robot is given the task of patrolling the corridor, maintaining line-of-sight with the base station at the entrance of the building (everything must be reported to or through the base station) while watching for potential enemies, while other robots explore the rooms on both sides of the corridor. In the following, we will show how the robots can help each other while accomplishing their respective tasks.

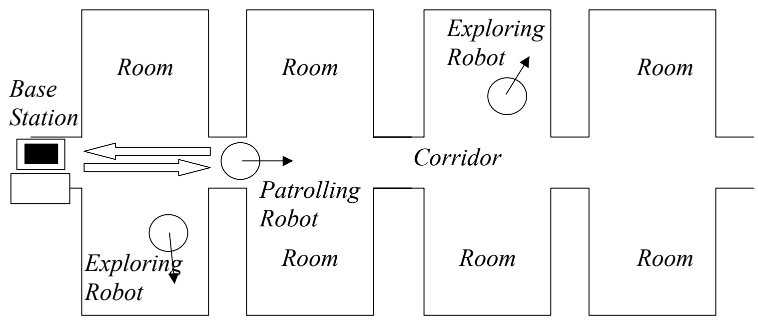


Figure 1. One robot is given the task of patrolling the corridor; other robots explore the rooms on both sides of the corridor.

2 THE NAVIGATION PROBLEM

In the above scenario (autonomous navigation in an unknown, potentially hostile environment), we define a navigation problem for a robot R_i as follows:

- R_i is given a starting point $Start_i$ and a set of goals $<G_{i1,}, \ldots, G_{iN}>$ to be reached in sequence. Goals can be both spatial location and interesting objects to be found, whose location is not necessarily known to R_i. When goal positions $<G_{i1,}, \ldots, G_{iN}>$ are known, they are expressed as x, y coordinates with respect to a relative reference frame F_i that is centered on R_i: we call $G_{ik,i}$ the kth goal position with respect to F_i. The notations R_i, $Start_i$, and G_{ik} are used to indicate the corresponding quantities with respect to a hypothetical absolute reference frame common to all robots that is used for the sake of simplicity in the following explanation. This absolute reference frame is not known nor used by the robots themselves. Finally, during each interval [t_{ik}, t_{ik+1}] only one goal G_{ik} is significant for the robot (we say that $G_{icurr} = G_{ik}$), while the others are ignored. If at time t_{ik+1} R_i reaches G_{ik}, we set $G_{icurr} = G_{ik+1}$. We say that, at time t_{ik+1}, R_i reaches G_{ik} if dist(R_i , G_{ik})<ε, where ε is a positive number that establishes a tolerance margin.

- R_i is provided with a local navigation algorithm *A* which returns, at each computation step t, what is considered as the best choice for reaching the goal G_{icurr}. The algorithm fails whenever it is not able to guide the robot to its current goal G_{icurr} (or to prove the non-existence of a path) in a finite time.

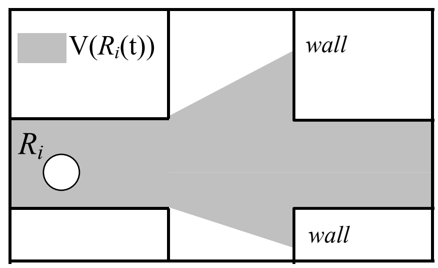


Figure 2. V(R_i(t)) is the visibility region of R_i (t).

- Suppose now that the free space F is a simply connected set with polygonal boundaries. The boundary of F is a finite set of closed, piecewise-smooth curves with only a finite number of non-smooth points. For any $R_i(t) \in F$ (that is, for any configuration in the free space that R_i can assume), let V(R_i(t)) denote the set of all points $V \in F$ such that the line segment that joins R_i(t) and *V* does not intersect the boundary of F. We call V(R_i(t)) the visibility region (Figure 2). In the following we assume that each robot R_i(t) is given omnidirectional visibility, and the corresponding visibility region V(R_i(t)) is contained within a circular area of bounded radius. Moreover, we say that robot R_i *can see* R_j (another robot) or *G* (a goal) if R_j or $G \in V(R_i)$. Remember that, in our definition, a goal *G* can be either an interesting object or a location in free space.
- Whenever R_i *can see* its current goal G_{icurr}, it starts heading towards it, by suppressing the output of algorithm *A*.

In the following we consider only the case in which the robot's goals coincide with a set of targets spatially located in the environment, whilst we ignore the case (very frequent especially in high risk, hostile environments) in which the goal coincides with a particular object (or person) to be found and identified. Thus, in our case, robots need to have some kind of approximate knowledge about the environment, since they know approximately the areas of the environment that must be explored, even if they do not know the topology of the free-space and, therefore, are incapable of planning a path. However, if we want to explore the environment to search for something, it is sufficient to substitute algorithm *A* (which determines the motion strategy of the robot and has been introduced in the second point) with an algorithm for autonomous exploration (even a random-motion algorithm) without affecting the general strategy that will be proposed for multi-robot cooperation. About the fourth point, notice that R_i chooses what he thinks to be the fastest path to the goal (this is guaranteed in a static environment) by following a straight line, no matter if *A* proposes a different motion strategy.

3 AGENTS COMMUNICATING THEIR OWN GOALS: COMMUNICATION IS PERCEPTION.

Let us consider a team composed of 2 robots. We introduce a very simple strategy that allows robots to obtain the benefit of line-of-sight communication improving each robot's behavior. Suppose for the moment that R_1 is the only robot that is given a navigation problem and is therefore freely moving in the environment, while R_2 remains in a fixed position. Whenever R_1 comes in sight of R_2 (i.e. $R_2(t) \in V(R_1(t))$ and $R_1(t) \in V(R_2(t))$):

1. *Share my goal:* R_1 communicates to $R_2 \in V(R_1)$ its current goal G_{1curr}.
2. *Check visibility*: If R_2 knows G_{1curr}, it checks its visibility ($G_{1curr} \in V(R_2(t))$). It then communicates the result to R_1.
3. *Head to goal*: If $G_{1curr} \in V(R_2(t))$ and $R_2(t) \in V(R_1(t))$ we let R_1 infer that $G_{1curr} \in V(R_1(t))$; that is, we are introducing a transitive property in the definition of the visibility function *V(R)*. According to point 4 of the previous section, whenever R_1 *can see* its goal it starts heading towards it, by suppressing the output of the navigation algorithm *A*. However, since R_1 perceives its goal through R_2, it sets $G_{1curr} = R_2$ and consequently heads towards R_2 until it is eventually able to directly perceive its own goal (this is guaranteed in a static environment). Obviously, this is not always possible if the robots are real and therefore they can obstruct the visibility of each other. Suppose for example that G_{1curr} is inside a room and R_2 is blocking the door: in this case, R_1 and R_2 should implement some sort of coordination strategy in order to make possible for R_1 to directly perceive its target once it has joined R_2.

Notice that the behavior of R_1 (heading towards the robot that declares to be able to see its goal) can be described by interpreting R_2 as a mirror that reflects the image of R_1's goal. We can imagine that R_1 heads towards the reflected image (thinking that it is the real one) until it is able to directly perceive its goal, thereby recognizing the illusion. On one side, this metaphor allows us to emphasize the parallelism between local perception/communication which motivates the approach: the mirror can be thought as a very simple communication device, a signal repeater that changes the direction of the visual signal: but for an external observer, what is the difference between perceiving the imagined goal reflected on the mirror and the real goal? From the other side, it allows us to easily explain the behavior of R_1 when G_{1curr} is neither directly perceived by R_1 nor by R_2, but it can be seen by a j^{th} robot R_j which shares this information with R_{j-1} and so on until the fact is eventually bounced to R_2 and finally to R_1 (see Figure 3). We can imagine robots R_2 to R_j as reflecting mirrors that transmit to R_1 the image of its goal. R_1 starts heading towards the only goal that it can perceive (the one reflected in R_2), but as soon as another image is available which guarantees a shorter path to the goal (in the sense of the path traveled by light from G_{1curr} to R_1: this concept will be clarified in the following), it changes its target and heads towards it until it is eventually able to directly perceive its own goal. In order to distinguish these concepts (visibility-through-reflection) from traditional visibility $V(R_i(t))$, we will use a different function $V_R(R_i(t))$ that we call *reflected visibility*.

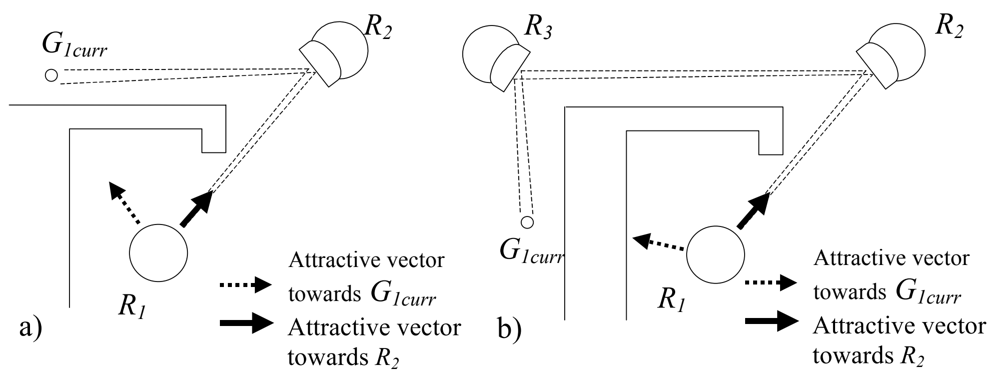


Figure 3. The behavior of R_1 when it *can see* its goal (a) reflected by R_2 or (b) reflected by R_2 and R_3.

In a static environment, this strategy will guarantee R_1 to finally reach its goal; however the trajectory generated is not optimum in terms of the distance that the robot has to travel. It is obviously possible to try to optimize the length of the path by considering the robots' visibility region and the geometry of the environment. However the simple behavior proposed offers another advantage: it allows the moving robot to keep its 'focus of attention' on the still robot from which it is receiving help. In many cases, this allows making visual processing faster and consequently increases the reactivity of the whole (the role played by attention in simplifying visual processing in human beings has been deeply investigated [9] [10]).

For a generic group composed of M robots, the rules for the behavior of each robot can be generalized as follows (once again we assume that R_1 is the only moving robot, whilst robots R_2 to R_M are stationary robots).

1. *Share my goal:* R_1 communicates to each robot $R_i \in V(R_1)$ its current goal G_{1curr}.

2. *Share other robots' goals:* If R_i knows G_{1curr}, it communicates the information to each robot $R_j \in V(R_i)$

3. *Check visibility:* If R_i knows G_{1curr} it checks its visibility ($G_{1curr} \in V(R_i)$). It communicates the result to each robot $R_j \in V(R_i)$.

4. *Reflect visibility:* If $G_{1curr} \in V_R(R_i(t))$ and $R_i(t) \in V(R_j(t))$ we let R_j infer that $G_{1curr} \in V_R(R_j(t))$; this is now a consequence of the transitivity of $V_R(R)$.

5. *Head to goal:* Once again, if $G_{1curr} \in V_R(R_i(t))$ and $R_i(t) \in V(R_1(t))$, then R_1 sets $G_{1curr} = R_i$ and consequently starts heading towards R_i until it is eventually able to perceive directly its own goal (guaranteed in a static environment). These rules will now be described in detail.

3.1 Rules 1 and 2: share goals with other robots

Since we ask the robots to share geometric information, we have to deal with the problem associated with the absence of a common reference frame (remember we are considering the case in which R_i's goals <$G_{,i1.,\ \ldots,}$ G_{iN}> correspond to known Cartesian coordinates in the free space F). It is usual in many multi-robot applications to assume that the robots share some kind of global reference frame, in which all the geometrical information exchanged between robots (if any) is computed (even in reactive approaches [11]). This assumption is indeed a very strong one, since the use of a global reference frame requires robots to be aware of their position and orientation in the world with high accuracy, at least if we want the information exchanged to have the same meaning for all the robots involved in the communication process. In the following, we describe a possible solution to this problem: similar solutions have been already presented in literature: see for example [12] (while robots move, one of them remains stationary thus acting as a landmark for the teammates) and [13] (a visual pattern is used in order to retrieve distance and orientation of teammate robots through visual sensing).

It is easy to show that, because of the line-of-sight communication constraint, the global reference frame assumption is not required. More precisely, suppose that the two robots R_1 and R_2 use different relative reference frames F_1 and F_2. Suppose also that, at time t, R_1 needs to communicate to R_2 its goal $G_{1curr,1}$. In order for robots to share meaningful information, R_2 needs to know the transformation matrix T_{21}(t) that permits the mapping of points between F_1 and F_2, as in computing $G_{1curr,2} = T_{21}(t)\ G_{1curr,1}$. Notice that T_{21}(t) is different from time to time, since robots are moving in the environment. Notice also that, for R_2 to compute T_{21}(t), it should be able to recognize R_1 's position and orientation with respect to its own reference frame F_2; however, while recognizing the position of a known object is a quite simple task (especially if the object to be recognized is designed in such a way as to be easily detected) the same is not necessarily true for its orientation. Moreover, it is well known that a wrong estimate of the rotational component of T_{21}(t) can be a source of significant errors.

The task is much simpler if we allow R_1 and R_2 to cooperate in computing T_{21}(t) (and T_{12}(t)). In fact, if R_1 is able to perceive R_2's position (x and y coordinates with respect to F_1) and R_2 is able to perceive R_1's position (x and y coordinates with respect to F_2), both robots are allowed to compute very accurately T_{21}(t) and T_{12}(t) by sharing information about the direction in which they see each other (see Figure 4a: α_{21} measures the direction in which R_1 sees R_2, α_{12} the direction in which R_2 sees R_1). From geometrical considerations (Figure 4b), it is straightforward to see that R_1 is allowed to compute θ_{21} (orientation of R_2 with respect to F_1) as $180+\alpha_{21}-\alpha_{12}$ and R_2 is allowed to compute θ_{12} (orientation of R_1 with respect to F_2) as $180+\alpha_{12}-\alpha_{21}$.

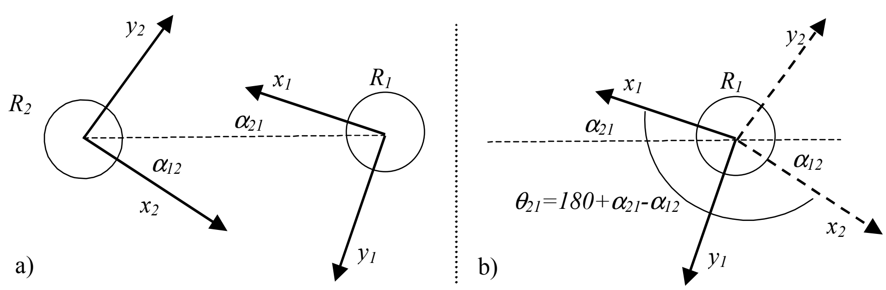


Figure 4. Robots cooperate to compute the transformation matrices T_{21}(t) and T_{12}(t).

Notice that, since the robots do not use a global reference frame, only R_1 is responsible for knowing with accuracy its own position with respect to the place where it needs to go. Even if R_2 is completely lost in the environment, it is still fully functional in helping R_1 to reach its target if R_1 asks it the right question (roughly speaking, we can imagine R_1 asking R_2: can you see my goal? It should be two meters on your left…). Thus, whenever robot R_1 communicates G_{1curr} to R_i, it sends the following data:

1. The goal position $G_{1curr,1}$ with respect to F_1
2. The angle α_{2i}, corresponding to the direction in which R_1 sees R_i
3. $Id_{1k} = k*M$, an identifier which unequivocally identifies R_1's kth goal.

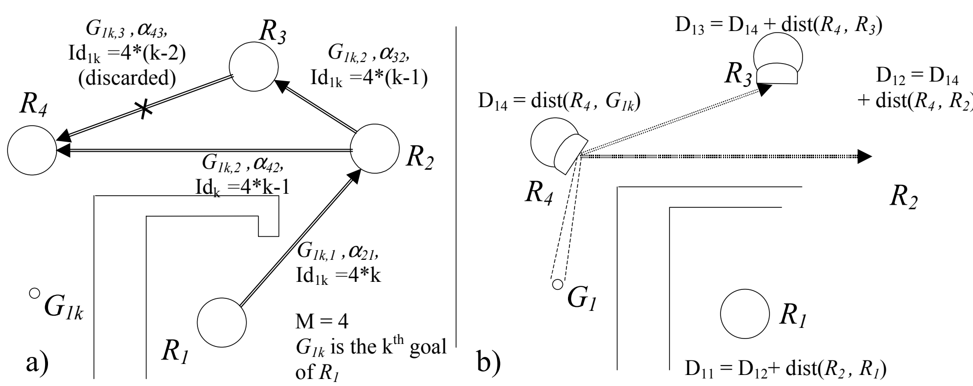


Figure 5. R_4 receives multiple reports *of* G_{1k} (from R_2 and R_3), and discards the one with the lower value of Id_k.

The identifier is required since G_{1curr} changes with time: whenever robot R_i receives G_{1curr}, it stores the goal (after having computed its position with respect to F_i) together with its identifier Id_{1k}. Next, when communicating it to another robot R_j, it marks G_{1curr} with Id_{1k} -1. R_j is thus aware of the fact that it has not received G_{1curr} directly from R_1, but from another robot that directly communicates with R_1. Notice that the identifier received by R_j still identifies unequivocally R_1's kth goal (since $k = 1 + (Id_{1k} - 1) / M$) but, on the other side, it also provides an estimate of the quality of the information, since it measures the number of robots by which the information has been manipulated. This is important because the computation of the transformation matrix is affected by noise in reality, and it is thus possible that R_j checks the position of G_{1curr} in the wrong place. To reduce the effects of noise, each robot will prefer to receive G_{1curr} directly from R_1 whenever possible. The number of coordinates transformations that have been performed on G_{1curr} before the information reached R_j can be computed as $Nr_{ik} = (Id_{1k} - (k - 1)*M)$, where $Nr_{ik} = M$ only if the goal has been directly communicated by R_1, and $Nr_{ik} = 1$ if the information has been manipulated by all the robots before reaching R_i. If R_i receives multiple reports about G_{1curr}, it accepts the new goal as a valid one only if its identifier is greater than or equal to the one that had been previously stored for that robot's goal (Figure 5a). That is, if the goal is more recent (updated information) or if it comes from a robot which is closer to R_1 (better quality information).

3.2 Rules 3 and 4: check goals' visibility and reflect visibility.

Each robot R_i that knows G_{1curr} checks its visibility ($G_{1curr} \in V(R_i)$); furthermore, it communicates the result to each robot $R_j \in V(R_i)$. Specifically, when R_i sees G_{1curr} it transmits the following information:

1. Id_{1k} = the identifier of the goal that it has seen (which can differ from R_1's current goal because of delays in inter-robot communication).
2. D_{1i}, the Euclidean distance between R_i and G_{1curr}.

We have stated that, if $G_{1curr} \in V(R_i(t))$ and $R_i(t) \in V(R_j(t))$, we let R_j infer that $G_{1curr} \in V_R(R_j(t))$. However, when R_j wants to communicate to another robot that it *can see* R_1's goal, it computes its distance to G_{1curr} as D_{1j} = dist (R_i, R_j) + D_{1i} which is clearly larger that the distance computed by R_i, the robot that can directly perceive the goal.

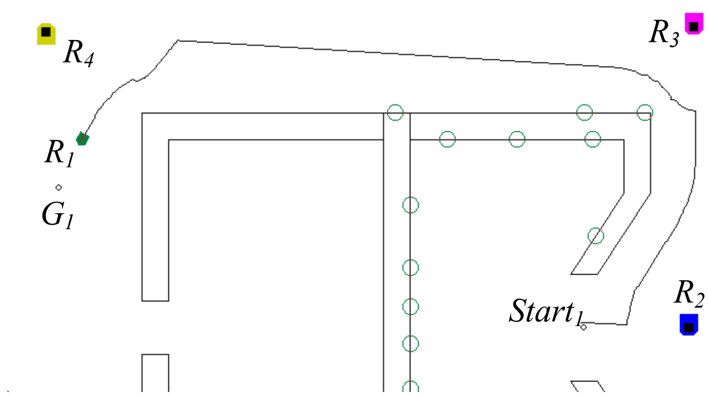


Figure 6. R_1's trajectory is shown (in simulation) when it receives information about its goal from R_2, R_3, and R_4

More generally when R_j receives information from different robots, each declaring to see G_{1curr}, it listens only to the robot that guarantees the shortest path to the goal, thus avoiding loops in communication (see Figure 5b). Next, when re-transmitting the information, it computes its distance to the goal according to the previous rule. Finally, R_1

accepts only the information that is associated with the shortest distance and discards the others, therefore heading towards the robot that provides the shortest path to its goal (guaranteed if the environment is static).

As a consequence, if suddenly R_1 perceives its own goal directly, it starts heading toward it since no other information can have a lower value as it is guaranteed to be the shortest distance. Figure 6 shows the behavior of the system for a very simple simulated case.

4 MOVING ROBOTS AND 'GHOST ROBOTS'.

Up to now we suppose a team of M robots of which only R_1 was given a navigation problem and therefore requested to move in the environment, while the other robots (R_2 to R_M) were stationary. One could try to compute, for a given environment, the minimum number of fixed support robots that is required to guarantee, for each pair of points in the environment (say R_1 and G_{1curr}), that the following expression is true: $G_{1curr} \in V_R(R_1(t))$. The problem seems very similar to coverage problems that have been deeply studied in the literature: art gallery problems [14] or the pursuit-evasion problem [15]. However, neither of these approaches fits very well into the motivation of our research. First of all, we aim at designing cooperation strategies for robots that have a very limited knowledge (or no knowledge at all) of the environment and therefore rely on local navigation algorithms to find their way to the target. Since the environment is unknown, it is not possible to compute the minimum number of fixed support robot that are required to fully cover the environment, nor to determine a motion strategy that allows in different times to displace M-1 support robots such as to make R_1's current goal visible from R_1's current position. Furthermore, even if computable, this number could be very high, and it appears to waste resources by using those robots only to help one robot to safely carry out its tasks.

Thus, we choose a different approach, and let *all* robots move in the environment while carrying out their own activities; that is, instead of determining a motion strategy or a spatial distribution that allow robots to help each other, we focus our attention on finding a strategy that allows them to do their best in helping each other whenever they have a chance. However, if we let the robots move without coordinating them, it is intuitive that the probability for the expression $G_{1curr} \in V_R(R_1(t))$ to be true becomes very low (and depends on the complexity of the environment, the number of robots involved, and the task of each robot). Thus, we choose to introduce a new visibility function, which we call *delayed visibility* $V_D(R)$. We wish to anticipate that, because of the following extension, we lose one of the properties of the previous approach, which was purely reactive in the sense that it did not require memory except for storing the two values Id_{ik} and D_i (the identifier of the goal and the length of the path), both required only for arbitrating between different solutions and choosing the optimum one. While previously we could imagine that at time t, R_1 perceived its own goal reflected by R_2, in the following this will no longer be true: R_1 still behaves as if it can see its own goal reflected in another robot which *has seen* its goal, but it is no longer guaranteed, even in a static environment, to reach its goal while following a finite set of straight trajectories. The algorithm can be described as follows (Rules 1 to 5 are the same as the previous case; we just need substituting $V_R(R)$ with the new function $V_D(R)$):

4.1 Rules 1 and 2: share goals with other robots.

Once again, when a generic robot R_1 shares its own k^{th} goal, it marks it with a unique identifier Id_{1k}, whose value is computed as in the previous case. Each robot R_j accepts G_{1k} as a valid goal only if its identifier Id_{1k} is greater than or equal to the one that it had been previously stored for R_1. Notice that now each robot needs to remember many different goals (a maximum of M-1 goals) while it is moving in the environment towards its own goal, while previously this was not needed since each robot received continuously updated information from all the robots that it could see. However this appears not to be a major problem, at least from the computational complexity and the memory requirements (both being linear with respect to the number of robots).

4.2 Rules 3 and 4: check goals' visibility and reflect visibility.

When R_j communicates to R_i that $G_{1curr} \in V_D(R_j)$, it transmits the identifier of the goal Id_{1k} together with D_{1j} (however, because of the introduction of $V_D(R)$, the estimate of the distance to G_{1curr} is now different; it will be described in the following paragraph). Thus R_i estimates its distance to the goal as $D_{1i} = \text{dist}(R_i, R_j) + D_{1j}$ and stores this value in its memory. If at a given time a third robot R_k comes in sight and communicates that $G_{1curr} \in V_D(R_k)$, R_i compares the stored distance D_{1i} with $\text{dist}(R_i, R_k) + D_{1k}$ and attends to the new information only if it guarantees a shorter path to the goal. Finally, R_1 accepts only the information that is associated to the shortest distance and discards the rest, heading towards the robot that provides the shortest path to its goal.

However, the same rule is now true even when the information $G_{1curr} \in V_D(R_i)$ is communicated by the same robot R_i, but at two different times (which happens frequently since robots are navigating in the environment towards their own goals). Suppose that at time t_1 robot R_2 provides a path with distance $D_{12}(t_1)$ (Figure 7). R_1 stores in its memory $D_{11}(t_1)=dist(R_1(t_1),R_2(t_1))+ D_{12}(t_1)$. If at time t_2 ($t_2 > t_1$) R_2 is farther both from the goal and from R_1, it provides a path which is given a score $D_{12}(t_2)$ such that $dist(R_1(t_2),R_2(t_2))+ D_{12}(t_2) > D_{11}(t_1)$. As a consequence, R_1 continues to head toward the location in which R_2 was at time t_1, even if R_2 is no more there. We can imagine that it continues to head toward a 'ghost robot' that provides the shortest path to the goal (as an aside: the ancient Greeks used the same word for 'ghost' and 'mirror', *eidolon*). Notice that this reduces the reactivity of the system, since R_1 now needs to remember the position $R_2(t_1)$ in which R_2 was located when it promised the best path, instead of being purely controlled by its own sensory inputs. This seems to be necessary whenever we allow robots to move, since otherwise R_1 would have the tendency to follow R_2 even if it is heading in the opposite direction with respect to G_{1curr}. Moreover, if R_1 is able to distinguish some interesting features in the environment and to use such cues for navigating towards $R_2(t_1)$, the consideration regarding keeping the focus of attention are still valid.

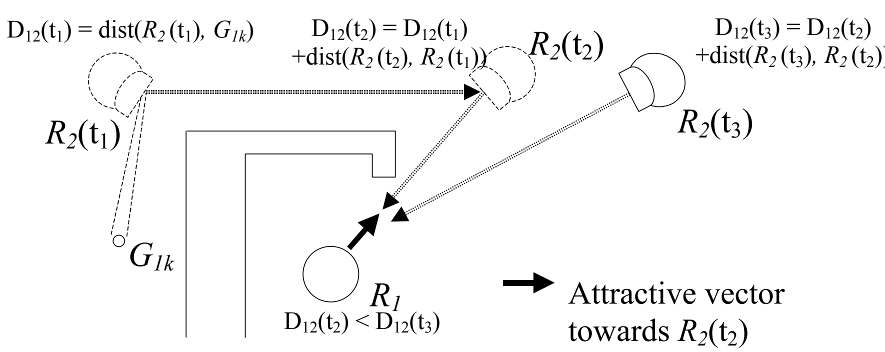


Figure 7. $R_2(t_2)$ provides a shorter path than $R_2(t_3)$. As a consequence, R_1 continues to head toward the location in which R_2 was at time t_2, even if R_2 is no longer there.

Finally notice that, since R_1 heads towards R_2 even if R_2 is not currently seeing G_{1curr} , R_1 is no longer guaranteed to perceive the goal (or finding further help for reaching the goal) when it has reached R_2's position. To the contrary, in most cases, it will have to start again to explore the environment by relying on its own navigation algorithm *A*, possibly from a 'better position'. We will attempt to demonstrate experimentally that this position is really a better one.

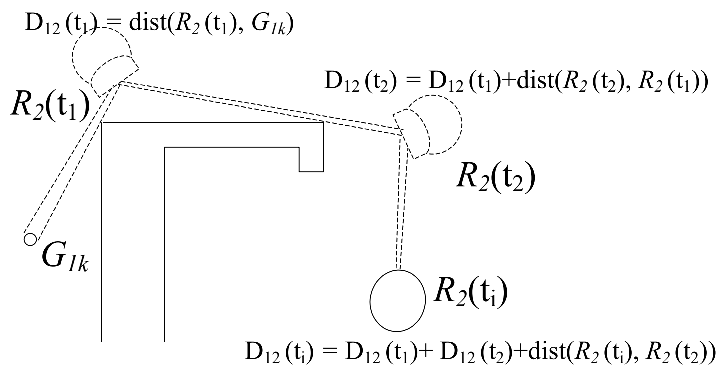


Figure 8. Computation of the distance to the goal when R_2 is moving.

4.3 Computation of the distance from the goal

Since R_i is moving, and not constantly in sight of G_{1curr}, we need to explain how it computes the distance D_{1i} when it communicates $G_{1curr} \in V_D(R_i)$. When R_i directly perceives G_{1curr} (i.e., $G_{1curr} \in V(R_i)$), D_{1i} is computed at each time step as dist(R_i, G_{1curr}) as usual; however, when G_{1curr} is no longer visible R_i recalls the last position $R_i(t_1)$ from which G_{1curr} was visible and computes $D_{1i}(t_1)$ = dist($R_i(t_1)$, G_{1curr}) and stores the result in memory (Figure 8). In the following, R_i checks if $R_i(t_1) \in V(R_i)$ and computes D_{1i} as $D_{1i}(t_1)$ +dist(R_i, $R_i(t_1)$). We can again imagine a 'ghost robot' $R_i(t_1)$ which sees the goal and transmits the location to R_i, which can then estimate its own distance to the goal (Figure 8). If, at time t_2 ($t_2 > t_1$) a $R_i(t_1)$ is also no longer visible, $R_i(t_1)$ and $D_{1i}(t_1)$ are substituted in memory with $R_i(t_2)$ and $D_{1i}(t_2)$, and R_i computes D_{1i} as $D_{1i}(t_2)$ +dist(R_i, $R_i(t_2)$).

4.4 Complexity of the algorithm

If we consider a team composed of M robots, each robot R_j needs to remember a maximum of M goals or 'ghost robots' together with the associated Id_{ik} and D_{ij} (corresponding to R_j's goal plus the goals of all the robots that it has seen) and to periodically check their visibility. This guarantees linear complexity and memory occupancy for the algorithm.

Whenever two robots meet, a communication mechanism with a bandwidth of order O(M) is needed since each of the two robots communicates a maximum of M terms (G_{ik}, α, Id_{ik}) and a maximum of M pairs (Id_{ik}, D_{ik}). If M robots are in sight and communicate with each other at the same time, the bandwidth required is $O(M^2)$.

5 AGENTS COMMUNICATING THEIR INTERNAL STATE.

In [8] a formal analysis is presented regarding the benefits provided by different types of communication in improving agents' behavior in different multi-agent scenarios. More specifically, agents explicitly sharing their goals and internal state are compared with agents making use only of implicit communication in three basic tasks (foraging, consuming, and grazing) that can be taken as the basis for building more general and complex tasks. One motivation is to provide a guide for the design of multi-agent systems, and to verify if the use of a more complex and therefore expensive language (required for sharing goals) is really more effective than communicating only one bit of information to encode the internal state, (analogous to what biologists call display behavior) or by avoiding any explicit communication altogether. Following these design guidelines, we provide robots with another mechanism for helping each other to achieve their own goal, which relies on the communication of their internal states encoded through a single bit of information. We want to verify if it is possible to use this different mechanism (the communication of the internal state) to help robots succeed in the cases where the previous mechanism fails and, more generally, to improve the overall performance of the system.

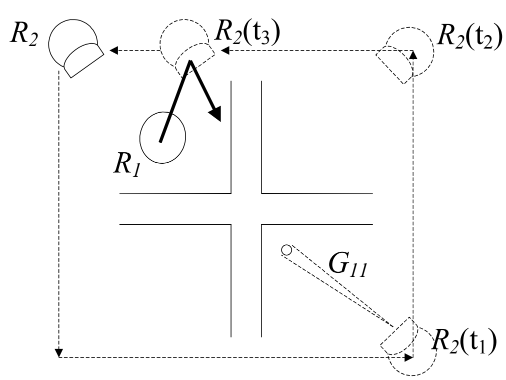


Figure 9. Once R_1 has reached $R_2(t_3)$, it is not in a better position for finding a path to its goal, and gets trapped again in the local minimum.

A typical case where the goal-sharing mechanism, presented earlier in this paper, fails appears in Figure 9. R_1's goal is to reach G_{11} , while R_2 is simply patrolling the area. Since R_2 has seen G_{11} when in $R_2(t_1)$, R_1 heads towards $R_2(t_3)$. However, once R_1 has reached $R_2(t_3)$, it is no longer in a better position for finding a path to its goal, and gets trapped into the local minimum.

The new mechanism proposed is not dependent on the local algorithm A that each robot uses for finding a path to the goal. Instead, the following strategy works with all generic algorithms A for which it is possible to define an estimate of its performance in conducting the robot to the goal. In a simple potential field approach, for example, it is easy for the robot to evaluate its progress to the goal and to realize when it is stuck in a local minimum (although it is not that simple to provide the robot with a strategy for escaping the local minimum!). Obviously, depending on the particular algorithm adopted, different metrics can be chosen for synthesizing the concept of 'being in trouble'. However, a very simple one can be chosen independently from the navigational algorithm adopted: if the robot is taking too much time in reaching its goal, it is very likely that the robot is having 'serious trouble' in achieving its mission.

With respect to the previous case, the rules required for robot cooperation are even simpler:

1. *Evaluate my progress:* Each robot stores a Boolean variable Tr_i: Tr_i is *false* if the robot thinks that it is progressing well towards the goal (the robot is *'not in trouble'*), *true* if it realizes that it is not making progress (the robot is *'in trouble'*).

2. *Share state*: Each robot R_i communicates to each robot $R_j \in V(R_i)$ its current state Tr_i.

3. *Head to helper*: If robot R_i is 'in trouble' and $R_j \in V(R_i)$ is 'not in trouble', R_j temporarily becomes R_i goal; that is, R_i starts heading towards R_j until it realizes that it is no longer '*in trouble*' or it is eventually able to perceive its own goal.

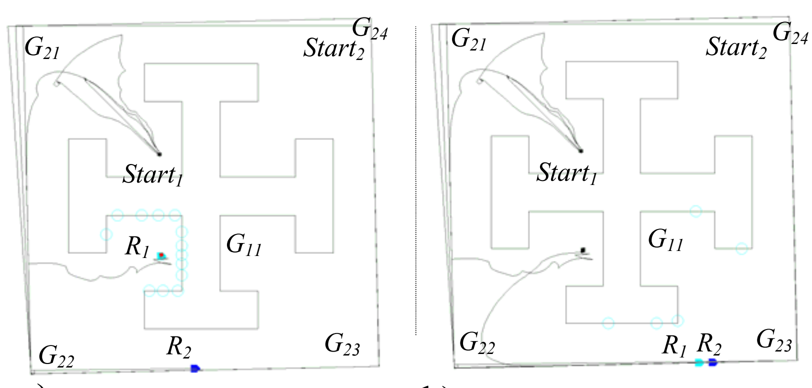
Once again, R_i believes temporarily that R_j is in a better area of the environment for reaching its own goal, and therefore heads towards R_j to increase its own performance. Obviously, the fact R_j is making progress towards its own goal does not imply following R_j will help R_i to reach a better position. However, if R_j is a support robot that patrols the environment, experiments confirm that the previous assertion is true in many cases (an example is shown in Figure 10: it will be described in details in the following).

Particular attention should be paid to the updating rule of the variable Tr_i, which defines the state of the i^{th} robot. In order to update Tr_i , each robot stores in memory a pair of variables

1. Pr_i, which increases at each computation step of the algorithm if the robot is not making progress towards the goal; it decreases otherwise.
2. Cr_i, which establishes an upper limit on the value of Pr_i.

Tr_i changes to *true* when $Pr_i = Cr_i$ and changes back to *false* when $Pr_i = 0$. Moreover, each time that Tr_i changes to *false*, we increment a third value Nt_i (whose initial value is 1), which counts the number of transitions between the state '*in trouble*' and the state '*not in trouble*' that have taken place in the robot. When the robot is not making progress towards the goal, Pr_i is increased by Nt_i; when the robot is progressing towards the goal, Pr_i is decreased by 1. Finally the upper bound to the value of Pr_i is set to $Nt_i\ Cr_i$.

This allows the behavior depicted in Figure 10 (in a simulated environment). Since it has no direct means (in this case) of coming out of the local minimum, R_1 remains stuck in its start position $Start_1$ (we have seen that, even if it communicates its goal G_{11} to R_2, this does not provide any help). While R_1 is stuck, the value of Pr_i increases by $Nt_i=1$ at each computation step until it reaches the maximum value Cr_i; when R_2 (which is progressing towards its goal G_{22}) comes in sight, R_1 starts heading towards it since Tr_1 = *true* and Tr_2 = *false*. However, since R_1 is now progressing towards its goal (by setting $G_{icurr}=R_2$), Pr_1 decreases to zero, and Tr_1 changes to *false*; R_1 stops following R_2 and gets again trapped in the local minimum while heading towards G_{11}.



a) b)

Figure 10. Thanks to the help of R_2, R_1 finally manages to find a path to the goal.

When R_2 comes in sight again, Pr_i probably has a higher value, since it has been incremented with a higher delta (Nt_i =2) and the upper bound is set to a higher value (2 Cr_i). This time R_1 follows R_2 for a longer time before heading again towards its own goal. After several attempts, it can be seen that R_1 finally manages to make its path to the goal (Figure 10).

6 Experimental Results

The system has been extensively tested in the simulated MissionLab [16] environment. Experiments have been performed in two different scenarios (Figures 11 and 13). Both scenarios represent typical office-like indoor environment of different complexity: R_1 is given an exploration task (it has to reach target positions which are located in the middle of the rooms) whilst other robots are deployed in the environment with the purpose of patrolling the corridors and supporting R_1 in its task. In particular:

- R_1 is assigned *exploration tasks*, each corresponding to a set of 200 randomly chosen targets to be reached in sequence. The same task is executed by providing R_1 with different navigation algorithms A: 1) a standard potential field algorithm, 2) a potential field algorithm to which the Avoid Past schema [17] has been added, and 3) the VisBug21 algorithm [7]. Moreover, each navigation algorithm is used to execute 5 *exploration tasks*, by randomly choosing different sets of targets to be reached. This allows us to compute the *average number of motion steps* required by R_1 to explore the environment with a given navigation algorithm together with its *standard deviation*.
- In order to test the coordination mechanism, the *average number of motion steps* required is first computed when no external help is available; next, the same exploration task is executed by adding an increasing number of supporting robots (R_2, R_3, and R_4) into the environment with the purpose of helping R_1 to accomplish its mission

Notice that all the navigation algorithms considered are local algorithms, i.e. they do not rely on global a-priori information about the environment. In particular: 1) does not guarantee the robot to reach its target without the help of other robots because of the presence of local minima in the potential fields, 2) proves to be effective to help the robot to explore the environment and 3) is guaranteed to find a path to the goal (whenever it exists) even when no helping robots are present. Thus, in the case of 1), the graphs in Figures 12 and 14, do not show any result when only R_1 is present. For analogous reasons, in the case of 1) we implement both the state-sharing strategy described in the previous section and the goal-sharing strategy described in sections II to IV. On the opposite, in the case of 2) and 3) only the goal-sharing mechanism is required. Finally notice that, at each motion step, the distance traveled by the simulated robots is fixed, since only the direction of motion changes: as a consequence, the number of motion steps required to execute an exploration task is directly proportional to the length of the path followed during the task.

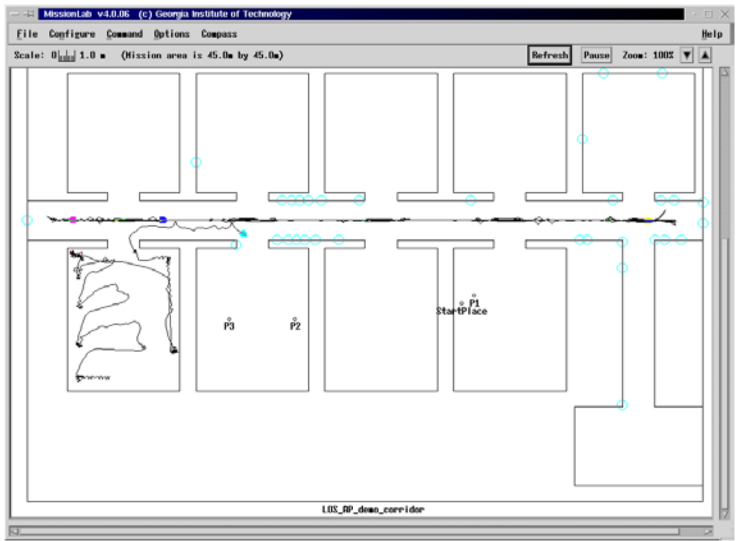


Figure 11. First simulated experimental scenario.

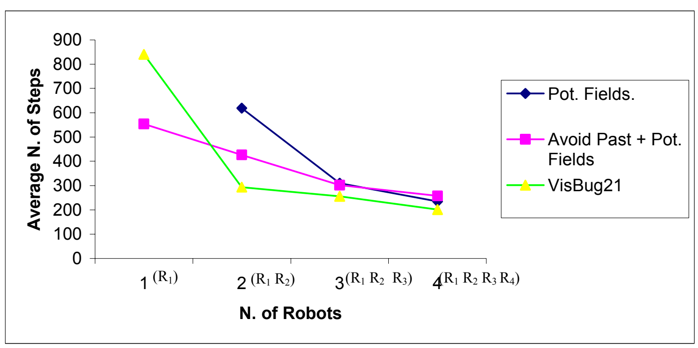


Figure 12. The results of the experiments performed in the first scenario.

In the two graphs in Figures 12 and 14 the *average number of motion steps* required by R_1 to complete exploration tasks is plotted versus the number of robot present in the environment: each curve corresponds to the different navigation algorithm. Notice that, in all cases, the performance significantly increases as we add more helping robots. In particular, in Figure 12 it can be seen that the number of motion steps required by the VisBug algorithm reduces from more than 800 to about 300 when adding just one support robot. Next, it decreases to about 250 when all three helping robots have been added. As regard to the other algorithms (potential field and Avoid Past), the improvement in performance is less evident in this very simple scenario: the number of computation steps decreases from about 550 to about 250 for Avoid Past and from about 630 to about 230 for the potential field algorithm. Table 1 shows in details the same results in fig. 12, by reporting the *standard deviation* corresponding to each value shown in the graph (*standard deviation* is expressed as a percentage of the *average number of motion steps* and provides a measure of the significance of the data).

Av.Past.	R1	R1,2	R1,2,3	R1,2,3,4
Av. Steps	554,2	426,2	302,0	257,2
St.Dev.	11,7%	8,1%	14,2%	5,6%
VisBug				
Av. Steps	840,6	294,0	255,4	201,2
St.Dev.	7,6%	6,6%	6,1%	3,7%
Pot. Fields				
Av. Steps		618,6	310,0	235,2
St.Dev.		3,6%	10,0%	5,4%

Table 1. Results of the experiments performed in the first scenario.

Finally notice that, in this scenario, the performance of the system seems not to depend on the particular algorithm adopted when we add more than two robots (the curves in fig.12 becomes very similar when N. of Robots>=3); since the environment is very simple, the trajectory followed by R_1 depends more on the paths suggested by R_2 R_3, and R_4 than on the algorithm *A*.

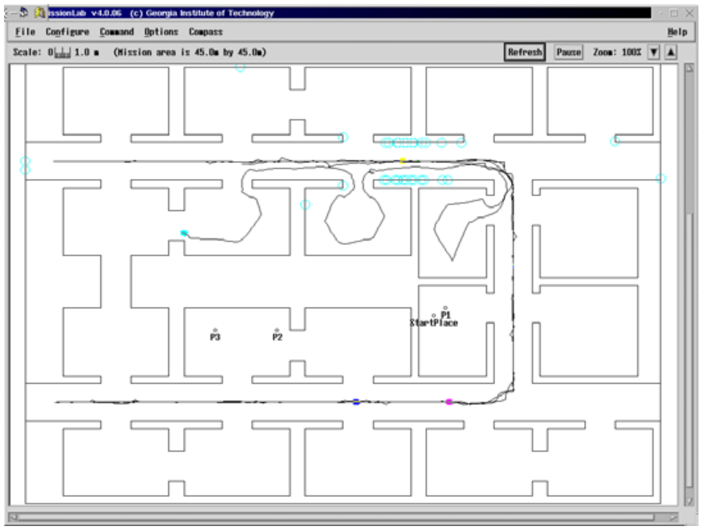


Figure 13. Second simulated experimental scenarios.

The situation is different in the second, more complex scenario, as it can be seen in Figure 14. Even when three helping robots are present, the influence of the navigation algorithm *A* is still important in determining the path followed by R_1, and consequently the number of computation steps required is different in the three cases. In particular, it can be noticed that VisBug is more efficient to find a path to the goal: the corresponding curve stays always below the two other curves, independently from the number of support robots which have been deployed in the environment. However, we are not interested in comparing the three algorithms, since their performance depends on the environment chosen and the tuning of some parameters. Instead, we want to show that the performance of a given algorithm improve when adding support robots: in this scenario, the number of steps required by VisBug decrease from about 500 to about 370, the steps required by Avoid Past decrease from about 1130 to about 680, and the steps required by the potential fields algorithm decrease from about 1250 to about 900. Table 2 shows in details the same results in fig. 14, by reporting the *standard deviation* corresponding to each value shown in the graph

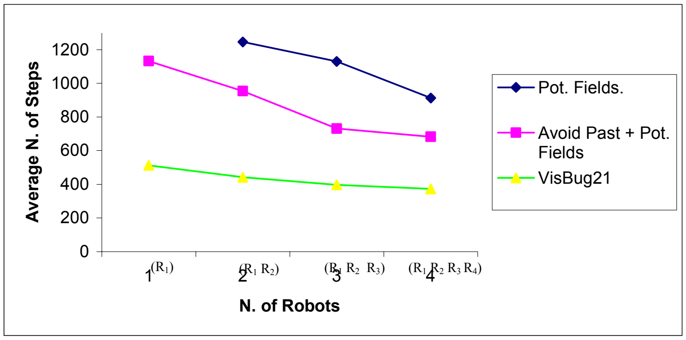


Figure 14. The results of the experiments performed in the second scenario.

Av.Past.	R1	R1,2	R1,2,3	R1,2,3,4
Av. Steps	1132,2	953,6	731,8	682,6
St.Dev.	4,5%	6,4%	7,9%	6,9%
VisBug				
Av. Steps	513,6	441,8	397,4	373,4
St.Dev.	10,5%	12,0%	9,0%	10,3%
Pot. Fields				
Av. Steps		1246,4	1130,4	913,6
St.Dev.		5,9%	6,7%	4,7%

Table 2 Results of the experiments performed in the second scenario.

In general, experiments show that the efficiency of all the algorithms is significantly improved when adding the simple coordination mechanism that has been presented in the previous Sections.

6.1 Adding noise to simulations

Up to now, we showed the performance of a system with perfect sensing and navigation capabilities: however, in a realistic implementation, it is required to deal with the problem of uncertain information. In general, we devise two main possible sources of uncertainty:

1. uncertainty in localization, i.e. each robot has only an approximate estimate of its position in the world
2. uncertainty in sensing, i.e. each robot has only an approximate estimate of the position of other robots which are in line-of-sight.

As regard to point 1, it should be noticed that uncertainty in localization is a general problem for navigation, and not a specific problem of the presented approach. Obviously, if robot R_1 wants to reach a target location in the environment, it has to know with a certain degree of accuracy its own position in the world with respect to a absolute reference frame; otherwise, it will reach a wrong location or, even, it will not be able to reach its target at all. The presented approach is not an exception. However, as it has been already explained in section III.A, only R_1 is responsible for knowing with accuracy the place where it needs to go; other robots, even if they are completely lost in the environment, are still fully functional in helping R_1, since they do never rely on the absolute reference frame to retrieve information about R_1's goal visibility.

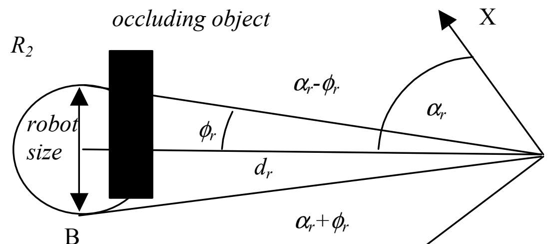
Thus, localization plays a role in the system only when robot R_1 is heading towards a 'ghost robot' (see Figure 7): if at time $t > t_2$ R_1 is heading towards the position which was occupied by R_2 at time t_2 (and, in the meanwhile, R_2 has moved), R_1 must rely on its own positioning system to reach $R_2(t_2)$. However, R_1 path is by definition a straight one, whose length is upper bounded by the radius of R_1's visibility area. Since the visibility radius is small in most indoor applications and the path is a straight one, we assume odometry to be adequate to guarantee $R_2(t_2)$ reachability. As a consequence of all this considerations, we choose to ignore localization uncertainty in our experiments.

As regard to point 2, we assume that each robot is equipped with a vision system, and it is therefore able to detect teammate robots in the environment (to make this task easier, one could put a well identifiable marker on each robot) and to retrieve teammates' positions by means, for example, of a inverse perspective mapping algorithm. Given this assumptions, uncertainty in sensing could be the consequence of two major causes: bad lighting conditions or partial occlusions. Since the effects of lighting conditions on a vision system are very complex to be modeled, we focus on the occlusion problem. In particular, when robot R_1 *sees* R_2, we assume that R_2's position with respect to F_1 (see Section III.A) is first computed in polar coordinates. Given R_2's real distance d_r and angle α_r, we define the perceived distance d_p and angle α_p

$$d_p = d_r + d_r d_{noise} \qquad (1)$$

$$\alpha_p = \alpha_r + \tan^{-1}\left(\frac{robotsize}{2d_r}\right)\alpha_{noise} \qquad (2)$$

d_{noise} is stochastic variable with a uniform distribution in the interval $[-err_{max}; +err_{max}]$, which introduces an error in the perceived distance: we assume that the maximum error in d_p depends on the real distance d_r, i.e. the farther is the robot, the bigger is the error. α_p is a stochastic variable with a uniform distribution in the interval [-1; +1]: notice that, since we assume that all errors are due to partial occlusions, we are allowed to put an upper bound on the maximum error on α_p (as defined in equation 2 and shown in Figure 15).



$$\phi_r = \tan^{-1}\left(\frac{robotsize}{2d_r}\right)$$

Figure 15. Because of a partial occlusion, *R_2's* angle α_p differs from the real angle α_r.

The experiments described in Figures 11 to 14 have been performed again after adding noise to the simulated vision sensor: Figures 16 and 17 correspond, respectively, to the environments in Figures. 11 and 13. The graphs show the *average number of motion steps* required by R_1 to reach its goals plotted versus err_{max} (when all the supporting robots R_2 , R_3, and R_4 are present). Finally, the rightmost value of each plot corresponds to the case in which no supporting robots are present.

Notice that, when err_{max} increases, the performance of the system decreases for every navigation algorithm. However, in Figure 16, even when err_{max} = 0.4 (corresponding to a maximum error which is about half the real distance d_r) the average number of steps required by R_1 to reach its goal is still lower than the case in which no supporting robots are present. On the opposite, in Figure 17 (corresponding to a more complex environment) it can be noticed that, when err_{max} increases, support robots become very soon almost useless. In particular, the performance of VisBug when err_{max} increases are even worse than the "R_1 alone" case.

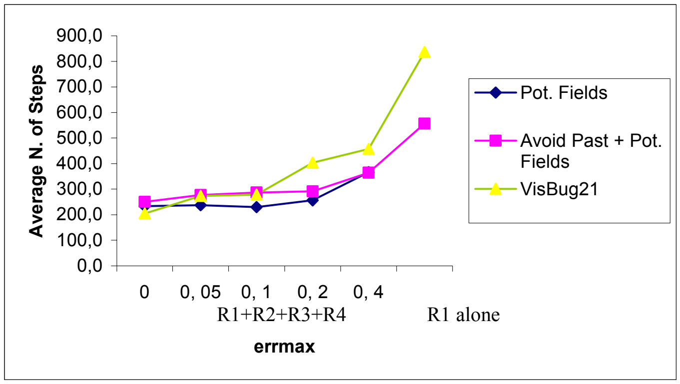


Figure 16. The results of the experiments performed in the first scenario (noise added)

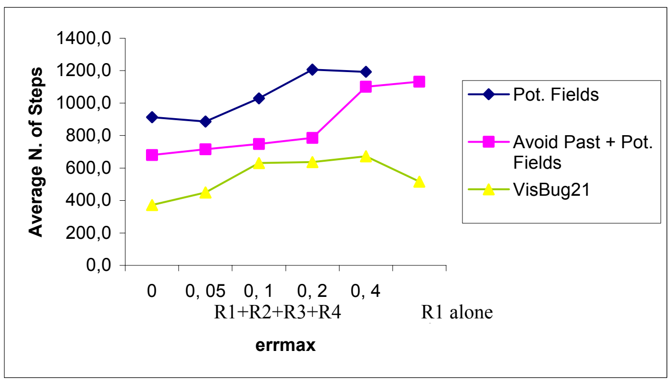


Figure 17. The results of the experiments performed in the second scenario (noise added)

7 CONCLUSIONS.

The work described in this paper deals with the problem of autonomous navigation and exploration in unknown indoor/outdoor scenarios (e.g. for high risk military applications). In this kind of scenario, a team of robots is deployed in order to minimize the time required for exploration and to maximize the coverage of a given area. In particular, we assume that each robot is assigned a set of goals to be achieved: goals, in our definition, can be both spatial location in the environment to be reached or particular objects to be found. The paper describes a simple (but effective) approach to the problem that allows robots to help each other in achieving their own goals when only line-of-sight communication is possible. The two strategies that have been implemented can be roughly summarized as follows:

1. *goal-sharing*: a robot is attracted by teammates that 'can see' or 'have seen' its goal.
2. *state-sharing*: a robot 'in trouble' is attracted by teammates that are 'not in trouble'.

Different experiments have been carried out in simulated environments, showing that individual robots increase their performance when allowed to cooperate with minimal communication/memory requirements. However, the experiments carried out up to the present time take into account only a subclass of the exploration problem as it has been defined at the beginning of this paper: i.e., we considered only the case in which goals correspond with spatial location to be reached, thus ignoring the case in which robots are looking for objects whose location is unknown. Finally, notice that performance increase more if we assign some robot the role of supporting other robot in their tasks. Each support robot can help more robots at the same time and that, as we pointed out in the introduction, patrolling robots are often required for different tasks, such as watching for the presence of possible intruders or maintaining line-of-sight communication with a fixed station located at one end of the corridor for transmission to the outside world.

Acknowledgments

This research has been supported in part by C.S. Draper Laboratory Inc. under contract #DL-H-521700.

References

[1] R.C. Arkin, 1989. "Motor Schema-Based Mobile Robot Navigation", *International Journal of Robotics Research*, Vol. 8, No. 4, August 1989, pp. 92-112.

[2] E. Gat, 1992. "Integrating planning and reacting in a heterogeneous asynchronous architecture for controlling real-world mobile robots", in AAAI-92, San Jose, CA, 1992, pages 809-815.

[3] J. M. Evans, 1995. "HelpMate: a Service Robot Success Story", *ServiceRobot: An International Journal*, Vol. 1, no.1, 1995, pp. 19-21.

[4] H. R. Everett, and D. W. Gage, 1996. "A Third Generation Security Robot", *SPIE Mobile Robot and Automated Vehicle Control Systems*, Boston, MA, 1996, Vol. 2903 pp. 20-21.

[5] J. Blitch, 1999. "Tactical Mobile Robots for Complex Urban Environment," *Mobile Robots XIV*, Boston, MA, September 1999, pp. 116-128.

[6] Y. Koren and J. Borenstein, 1991. "Potential Field Methods and Their Inherent Limitations for Mobile Robot Navigation", *Proceedings of the IEEE International Conference on Robotics and Automation*, Sacramento, California, April 7-12, 1991, pp. 1398-1404.

[7] V.J. Lumelsky and T. Skewis, 1990. "Incorporating range sensing in the robot navigation function", IEEE Trans. On System Man and Cybernetics, 20(5) pp. 1058-1068, 1990

[8] T. Balch and R.C. Arkin, 1994. "Communication in Reactive Multiagent Robotic Systems", *Autonomous Robots*, Vol. 1, No. 1, 1994, pp. 27-52.

[9] L. Itti and C. Koch, 2000. "A saliency-based search mechanism for overt and covert shifts of visual attention", *Vision Research*, 40, 2000, pp. 1489-1506.

[10] B. Olshausen and C. Koch, 1995. "Selective Visual Attention," *in The Handbook of Brain Theory and Neural Networks*, ed. M. Arbib, MIT Press, Cambridge, MA, 1995, pp. 837-40.

[11] L.E. Parker, 1999, "Cooperative Robotics for Multi-Target Observation", *Intelligent Automation and Soft Computing*, special issue on Robotics Research at Oak Ridge National Laboratory, 5 (1), 1999, pp. 5-19.

[12] R. Kurazume and S. Hirose, Study on Cooperative Positioning System - Optimum Moving Strategies for CPS-III -, Proc.IEEE Int. Conf. On Robotics and Automation, Vol. 4, pp. 2896-2903, 1998.

[13] G. Dudek, M. Jenkin, E. Milios, D. Wilkes, Experiments in sensing and communication for robot convoy navigation-, Proceedings IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), Pittsburgh, PA, August, 1995, Vol. 2, pp. 268-273.

[14] J. O'Rourke, 1987. *Art Gallery Theorems and Algorithms*, Oxford University Press, New York, NY, 1987.

[15] L.J. Guibas, J.C. Latombe, S.M. LaValle, D. Lin, and R. Motwani, "A Visibility-Based Pursuit-Evasion Problem", *International Journal of Computational Geometry & Applications*, Vol. 9, Nos. 4 & 5 (1999) 471-493.

[16] R.C. Arkin, T.R. Collins, and Y. Endo, 1999 . "Tactical Mobile Robot Mission Specification and Execution", Mobile Robots XIV , Boston, MA, Sept. 1999, pp. 150-163.

[17] T. Balch, and R.C. Arkin, , 1993. "Avoiding the Past: A Simple but Effective Strategy for Reactive Navigation", *Proc. 1993 IEEE International Conference on Robotics and Automation*, Atlanta, GA, May 1993, Vol. 1, pp. 678-685.