Failure anticipation in pursuit-evasion

Cyril Robin*† and Simon Lacroix*†
*CNRS, LAAS, 7 avenue du colonel Roche, F-31400 Toulouse, France
†Univ de Toulouse, LAAS, F-31400 Toulouse, France
cyril.robin@laas.fr, simon.lacroix@laas.fr

***Abstract*—This paper presents a new approach for the pursuit of targets by a team of aerial and ground robots under realistic conditions. Mobile target pursuit is often a sub-task of more general scenarios, that call for environment exploration or monitoring activities. Since most of the time a single robot is sufficient to ensure the pursuit of a target, our approach aims at minimizing the team resources devoted to the pursuit: while ensuring the pursuit, a single pursuer evaluates its own potential failures on the basis of the situation defined by the target evolution and the environment structure. It thus assesses its need for team support. When support is necessary to keep the target in view, one or more additional robots are involved, according to a task allocation scheme. We provide mathematical bounds of the complexity of the approach, that ensure that the system has real-time performance. Simulations in a variety of realistic situations illustrate the efficiency of the proposed solution.**

I. Introduction – context

Numerous mobile robotics applications are related to "targets", be the target adversaries to detect and chase, flocks of animals to monitor, other robots to follow or assist... Detecting, localizing or tracking targets raises a large variety of problems, to which the research community has devoted a lot of work. Figure 1 summarizes the primary problems encountered in target related applications. One may first distinguish two different missions: detecting one or more targets, and tracking them.

The first mission aims to control an area, and ends with the detection or the capture of the targets inside this area, using several agents, robots or fixed sensors. The historical problem is known as the art gallery problem [16] which considers fixed sensors. More recent variations are the patrolling problem [14] and the surveillance problem which consider either mobile sensors only or a combination of mobile and fixed sensors. Besides, most problems known as pursuit-evasion or search problems are actually "capture" problems, where the aim is to surround a target, avoiding both the contamination of previously cleared areas and the evasion of the targets. It is assumed that the number of robots is sufficient for such purpose. Chung et al. [5] propose a good survey of existing mathematical and robotics oriented work in this area.

Discovering or detecting a target is one thing, but in real applications this is often only a part of the whole scenario, either because the robots cannot neutralize the targets or because it is not desired or expected. Once the targets are designated, the robots often have to track them. This is the second kind of mission, which starts upon target detection. One may then want to localize the targets [10] or to monitor them. Localization may use different sensors and vantage points [19] or some targets specificities like group coherence [17], while monitoring is achieved through a direct view to the targets. With several targets, the problem often comes to trade the targets between the observers, depending on their relative positions [7, 8]. When there is only one target, a single robot may perform the tracking task alone – which we call "following", but it often referred to as "pursuit-evasion". This latest problem often assumes that a single robot is enough to perform the target monitoring task.

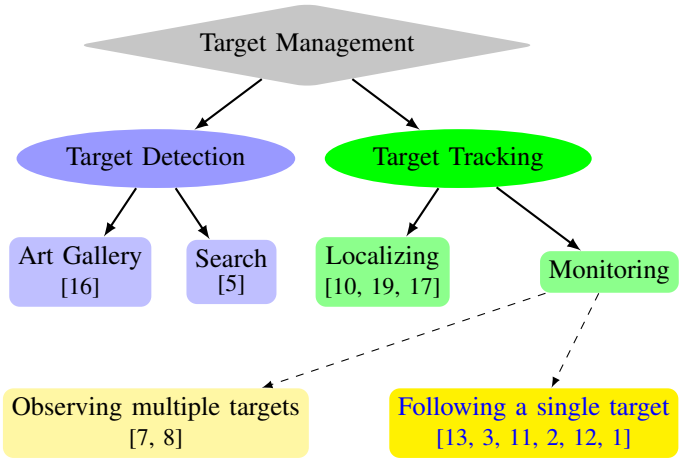


Fig. 1. Overview of the target management problems.

For all these problems, whatever the application context and the target type considered, multi-robot teams naturally extend the capacity of a single robot to manage the targets. Heterogeneous team provides even more solutions and opportunities to perform a mission, *e.g.* by multiplying the vantage points, as in cooperating aerial and ground teams. Also, while a single kind of robots can perform well on specific cases, a heterogeneous team can achieve multi-objective missions, some robots being more adapted to target tracking, others being better fitted to target detection for instance.

We hereby consider a multi-objective scenario where an heterogeneous team of robots is in charge of controlling a predefined known area. They are *not* numerous enough to statically cover the entire area (this is not an art gallery problem), and several targets can be present in the environment. The robots are initially engaged in an exploration phase, according to a pre-planned strategy. Once a mobile target is discovered, the team must ensure that the target remains visible, *while*

pursuing the environment exploration. The scenario naturally imposes the satisfaction of real-time constraints.

The paper focuses on the target pursuit phase, under the realistic constraints raised by this multi-objective scenario. The only hypothesis on the target is that it is not harmful for the robots: besides this it may either be adversarial, evasive, or move according to any other strategy. The success of the tracking task is defined according to some visibility criteria (in terms of distance and continuity, from a 100% target visibility constraint to a more relaxed one).

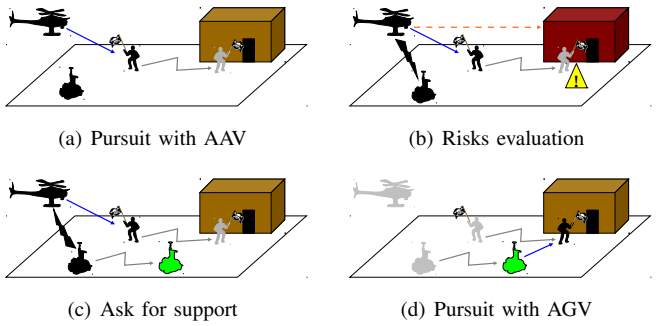
As the team performs both exploration and one or more tracking tasks, we want to minimize the resources required by the management of one target: the objective is to satisfy visibility constraints on the target, while minimizing the number of required robots for this purpose. As a single pursuer is most of the time sufficient to perform the tracking task, this is the favored tracking configuration. However a single pursuer can sometimes fail, in which case it is asks for support by other robots. The two main issues raised here and tackled in this paper are *(i)* how to predict the single pursuer failures and *(ii)* how to anticipate and prevent them with support robots.

The next section reviews the main results of the literature on target following, formalizes the problem, and presents our approach and the used models. Section III is the heart of the paper: it depicts how the pursuer can assess future tracking failures while the target is being tracked. Results in various realistic situation are presented and analyzed. Section IV exploits these potential tracking failures to define cooperative support tasks, using a task allocation scheme, and a discussion concludes the paper.

II. The pursuit problem

Much work on the target following problem can be found in the literature. Eaton and Zadeh [18] first exposed the search for an optimal strategy as an optimization problem in discrete probabilistic systems, working with huge search space size. More recently, contributions by Hutchinson et al. [13, 3, 11, 2] thoroughly analyze the problem and some of its variations: the following problem is fully decidable, but its complexity hinders the definition of optimal solutions under real time constraints. Besides, some single robot local following strategies however exhibit great results in both simulations and actual experiments (*e.g.* [12, 1]). Note that these approaches exploit poor environment models, usually 2D binary free/obstacle models in which the obstacle areas coincide with visibility masks.

To minimize the resources allocated to the target pursuit, our approach is to mainly rely on one robot (the *pursuer*) to perform the pursuit task: the state of the art shows that this should be sufficient most of the time. But due to its capacities, the target capacities and the environment configuration, the purser may fail to ensure the visibility constraint, be its strategy optimal or not. The pursuer therefore constantly evaluates the potential upcoming failures and their associated risk, and asks for support from others robots only when required. For instance "the target is rather far and about to enter a maze" or "the target is about to cross an area the pursuer can not cross" are situations that do call for support from other robots, whereas the situation "the target is moving behind a small building located in a wide open area" does not if temporary occlusions are allowed (figure 2).



(a) Pursuit with AAV (b) Risks evaluation

(c) Ask for support (d) Pursuit with AGV

Fig. 2. Illustration of our approach. In (a) the helicopter is the pursuer. In (b), the target is about to enter a building in which the helicopter can not proceed: the pursuer asks for support (c), and the ground robot becomes the pursuer (d).

A. Problem statement

According to an economy of means principle, we want to minimize the number of robots requisitioned for the tracking task.

Let R the set of robots. The problem is formally defined as follows:

$$\forall \tau, \text{minimize} \sum_{r \in R} a(r, \tau) \tag{1}$$

where

$$a(r,\tau) = \begin{cases} 1 & \text{if the robot } r \text{ is active at time } \tau \\ & \text{for tracking or support in pursuit} \\ 0 & \text{else} \end{cases} \tag{2}$$

while satisfying the visibility criteria :

$$\int_{\tau - T_{max}}^{\tau} h(\{r, a(r) = 1\}, target, \theta) d\theta \leq T_{max} \tag{3}$$

with

$$h(S, target, \theta) = \begin{cases} 1 & \text{if } target \text{ is hidden} \\ & \text{from the set of robots } S \\ & \text{at time } \theta \\ 0 & \text{else} \end{cases} \tag{4}$$

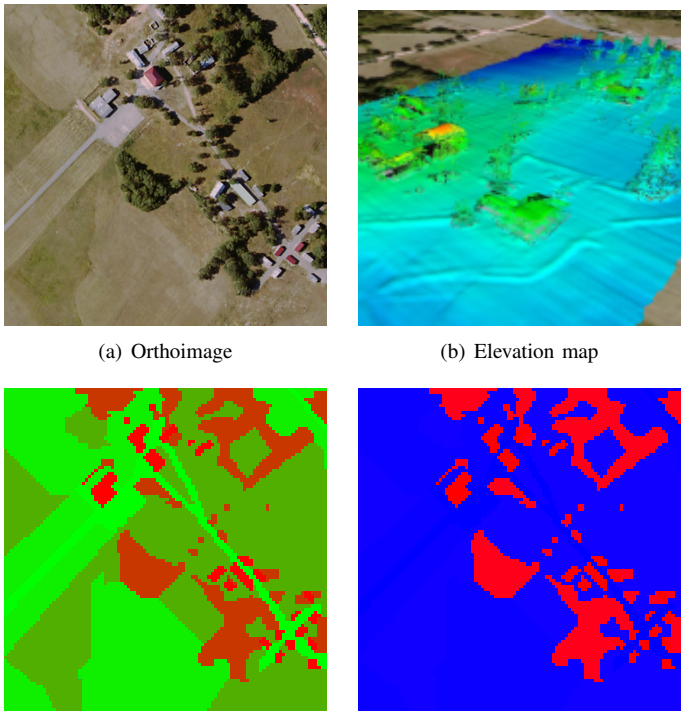
More specifically, the target is hidden if none of the active robots sees it (*i.e.* it is occluded by the environment or beyond the sensor maximal range). T_{max} is a criteria specified by the operator, which allows relaxing the visibility conditions: the target may be out of sight, but no longer than T_{max} seconds (T_{max} can also be set to 0).

B. Realistic Models

To be able to deal with a whole variety of realistic environments, different types of target and heterogeneous robots, we use environment models that explicit traversability properties for both the ground robots and the target and on which visibility constraints can be assessed.

A layered model of the environment allows to recover inter-visibility and traversability information (namely, a 2.5D elevation map and a symbolic layer that expresses the terrain type, *e.g.* roads, grass, obstacles, buildings, rivers...). Motion models are associated to each vehicle (robots and targets), they define their movement capacities in terms of attainable speeds.[1] The robot sensors are modeled as omnidirectional cameras, to which a maximal range of detection is associated.

By convolving the vehicle motion models with the terrain type layer, we end up with a multi-layered map for traversability, used for target motion prediction and robot planning. A priori target visibility information are also computed for each robot (*e.g.* an AAV will never be able to view a target in a building), and the 2.5D elevation map is exploited to assess visibilities (see figure 3). All these layers are represented by Cartesian grids, and are exploited online to compute the motion and sensing possibilities using the available information on the current state (such as target speed and position). Each robot embeds the models of its motion and sensing capacities, and the various target motion models.



(a) Orthoimage (b) Elevation map

(c) Traversability map for AGV (d) Visibility map for AGV

Fig. 3. Environment models used. (a) Satellite view of the area (orthoimage); (b) elevation model (derived from an airborne lidar survey); (c) 2.5D multi-channel traversability map: different colors correspond to different attainable speeds, along a green/red fast/slow scale; (d) 2.5D multi-channel visibility map.

III. Assessing the need for team support

Our approach relies on the fact that the robot which tracks the target, called the *pursuer*, is able to predict *tracking failures*, that is loss of target visibility for a duration longer than T_{max}. For this purpose, we evaluate all the possible pursuer / target situations over a temporal horizon h to isolate the tracking failures that can occur between the current time τ and the time $\tau + h$. We hereby depict how this can be achieved in real-time – the predicted failures generate support task requests, whose processing is depicted in section IV.

A. Failure evaluation

As failures mainly depend on the environment and on the relative positions of the target and the pursuer, there is no real general shortcut for their assessment, and so one has to compute all the possible future states over the time horizon h. A future state is a failure state if constraint (3) is not satisfied, which for the pursuer is equivalent to:

$$\forall \theta, \theta \in [\tau; \tau + T_{max}], h(\{pursuer\}, target, \theta) = 1 \qquad (5)$$

i.e. if the target is hidden from the pursuer sensors for T_{max} seconds or more.

To compute all the possible states, we exploit the discrete structure of the traversability models, and define a tree using the pursuer and target motion models[2]. At every time step, the tree grows with a branching factor of $m_t * m_r$, where m_t and m_r are respectively the number of possible motions for the target and the robot. With a temporal horizon h, the complexity of the tree building is $O((m_t * m_r)^h)$.

Usual values for the parameters are $m_t = m_r = 9$ (9-connexity in 2D-grid), and $h = 20$ at least. h must be large enough so that the other robots of the team can address the requests for support, without *a priori* constraining them too much to remain in the vicinity of the pursuer: indeed the smallest h is, the quicker and closer the supporting robots must be. Using a value of h of 20 and considering 9-connexity, the tree complexity is $O(81^h)$, *i.e.* near 10^{38}, which is by no means tractable.

B. Introducing the tracking strategy

Besides the combinatorial problem of the failures evaluation, the pursuer tracking strategy has to be efficient to minimize the needs for team support. *Efficient* stands here for "keep the target in sight", while being predictable and *fast to compute*. Indeed we seek to evaluate all the possible outcomes with unpredictable target motions, hence numerous strategies

[1] We assume that once detected, a target is identified – hence its capacities are known to the pursuer

[2] For the sake of simplicity, we consider throughout this section that the purser and the target evolve at the same speeds. This does not cause any loss of genericity, but only slightly influences the overall complexity – see appendix.

must be evaluated within the time horizon h to cope with this unpredictability.

As computing an optimal tracking strategy raises a combinatorial problem, we use a local greedy strategy. One may want to use a pre-computed optimal strategy but this would not be robust to differences between the *a priori* environment model and the actual environment. Real-time state-of-the-art local strategies are efficient, but still require much computing resources and often make strong assumption about the target motions. Our local strategy applies the two following rules: (1) if the target is visible, try to get closer to the target while maintaining visibility; (2) else, find the shortest path to the closest position which satisfies the target visibility criteria. While clearly sub-optimal, this simple local strategy shows acceptable tracking performances, is extremely quickly computed, and can be applied to either an AAV or an AGV.

This local strategy also brings an important advantage: the branching factor of the tree is directly reduced to m_t, the number of available positions for the target, because the strategy associates a single robot position for a given target position. The tree complexity becomes $O(m_t^h)$ (see figure 4), that is near 10^{19} with $m_t = 9$ and $h = 20$. Note that m_t could be reduced by making assumptions on the target motions, but this is not desired, and this does not drastically reduces the problem the exponential complexity.

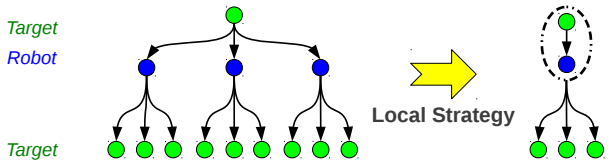


Fig. 4. Reducing the complexity: the impact of a local strategy

C. From a tree to a cyclic pursuit graph

The previous complexity gain is not enough to allow the satisfaction of real-time constraints: the complexity needs to be further reduced, while still ensuring the detection of all the potential failures. Various structure transformations of the tree allow to find and exploit states redundancies, that are brought by the fact that the tree is built upon a grid structure.

First, note that several positions p_t of the target may be handled by only one robot position p_r. The former state-nodes (τ, p_t, p_r) become new nodes of type $(\tau, \{p_t\}, p_r)$, with τ the considered time. This lets the tree structure unchanged (figure 5).

Second, for a same stage in the tree, there are several similar nodes which only differ by their parents. However the evaluation of a node only depends on spatial (and sometimes temporal) considerations, not on the past positions of the target or the robot. So similar nodes can be gathered, even if they have different parents. The tree structure can be changed into a graph structure, but the information carried by the nodes are the same (figure 6).

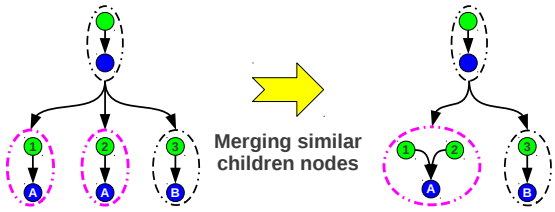


Fig. 5. Reducing the complexity: merging spatial redundancies

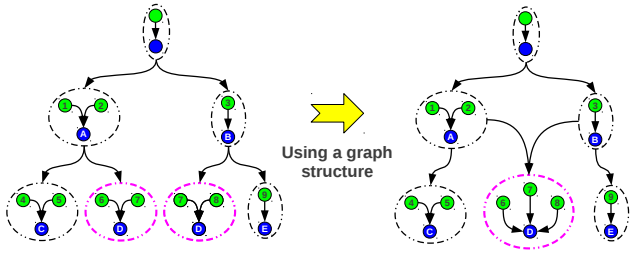


Fig. 6. Reducing the complexity: using a graph structure to exploit spatial redundancies

For a 9-connexity grid, these two structure transformations drastically reduce the complexity, down to $O(h^5)$ in the worst case (see appendix for details). Temporal redundancies allow to further reduce the complexity, by introducing temporal loops in the graph: some situations are indeed encountered several times, *e.g.* when none of the vehicle move, or after one loop around a building (figure 7). The nodes are embedded the following data: $(\hat{\tau}, \{p_t\}, p_r)$ where $\hat{\tau} = \min_\tau \{\tau / (\tau, \{p_t\}, p_r)\}$, *i.e.* $\hat{\tau}$ is the first temporal occurrence of the considered spatial state. The resulting structure is referred to as the *pursuit graph*.

Last, we introduce an iterative algorithm to build the graph, which allows to only consider the real new nodes at each next temporal step, which are referred to as the *front nodes*. This shrinks the complexity down to $O(h^3)$, still without any loss of information nor precision in the prediction (see appendix for details).

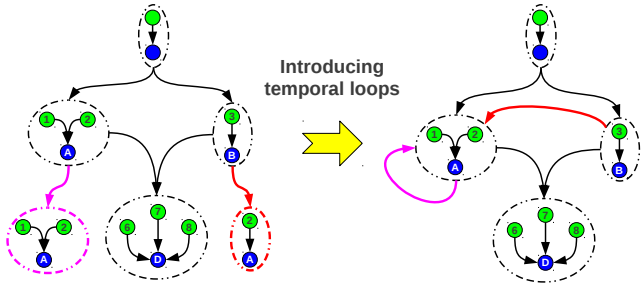


Fig. 7. Reducing the complexity: introducing temporal loops

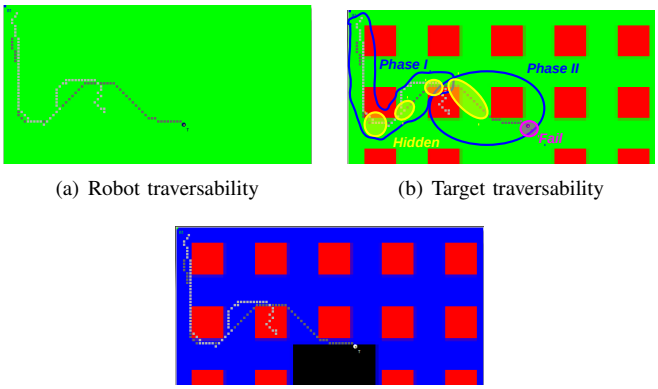
The resulting complexity is polynomial of low degree, which allows to assess the potential failures, and hence the needs for team support in real time. Each potential failure generates a task, which has the form (τ, p) where τ is the date

of the failure and p its position. In other words the support task is defined as *watch the position p at date τ*. The result is a small set of tasks, which is empty most of the time (when the target is fully under the control of the pursuer), and which constitute the seeds for multirobot cooperation.

D. Results

The bounded tractable complexity and the fact that the elaborated solution guarantees the assessment of all the potential failures are satisfying, but in practice, how does the pursuer perform and are the complexity bounds low enough for real-time applications?

Three examples are hereby presented to appreciate the performance of the solution: one takes place in a Manhattan-like environment, one is a "river-crossing" situation, and one exploits models from an actual experimental field.



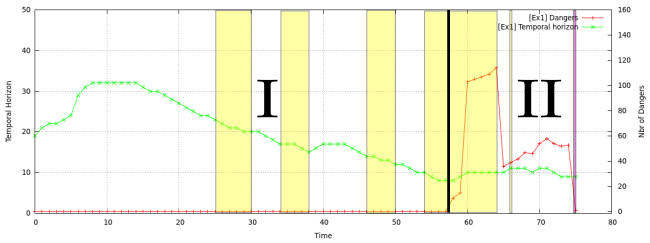
(a) Robot traversability (b) Target traversability

(c) Robot visibility

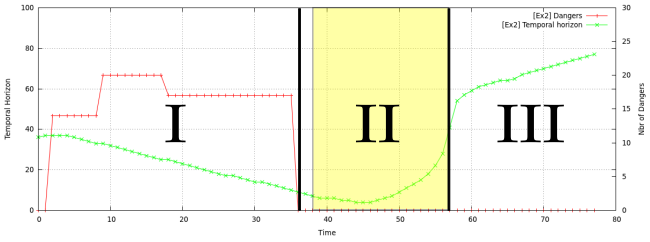
Fig. 8. Pursuit in a Manhattan environment: (a) traversability for the pursuer (AAV); (b) traversability for the target (ground target); (c) Visibility map: elevations of the buildings, and presence of a covered area in the bottom center of the map (in black). The vehicle trajectories are displayed (dark grey = target; light grey = pursuer), and the yellowish areas indicates places where the target is temporary hidden. The target escapes in the covered area (which is a definite failure, denoted as *fail*). In phase I the target is under control, while in phase II the target is too close to the covered area (danger).

1) Manhattan situation: Figure 8 shows the traversability and visibility maps, and the trajectories of both the target and the robot pursuer. The situation is a typical pursuit in a Manhattan-like town center, with a ground target and an AAV pursuer. There are buildings and, in the middle, a covered area where the AAV can not enter in nor see through (it can only fly above). The buildings do not constraint the pursuer movement, but its visibility. The motions of both the target and pursuer are holonomic.

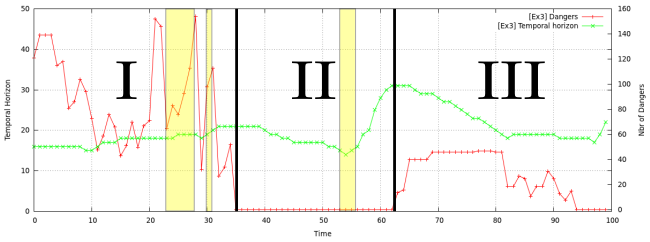
Figure 9-a displays the number of *dangers* (in red) and temporal horizon (in green) over the time. Dangers are states that do require a request for support from other robots of the team. One can distinguish two phases: in phase I the target is under control (no predicted failures), even if the target is sometimes temporary hidden (in yellow). The temporal horizon increases at the beginning (easy environment) and then lowers as more buildings obstruct the visibility. This is the result of the real-time constraints and the adjustment of the



(a) Manhattan environment



(b) River-crossing situation



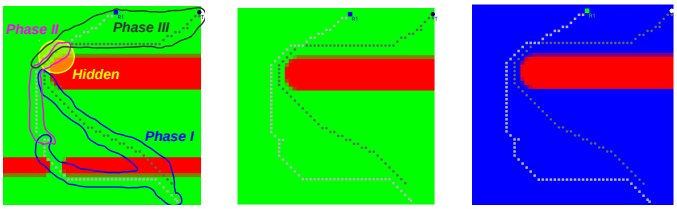
(c) Realistic Environment

Fig. 9. Evolution of the number of dangers (in red) and of the temporal horizon (in green) as a function of time steps for the three considered cases. The time laps during which the target is temporary hidden are highlighted in yellow.

computation time required by the building of the pursuit graph (see paragraph III-D4). In phase II, the target approaches the covered area, which generates several danger states – and in the absence of support by an other robot, the target eventually enters the covered area.

The figures highlight that the greedy strategy, although not optimal, behaves quite well. Furthermore, the system generates temporally and spatially gathered tasks (remind each danger generates one task). This is really important because on the one hand it will avoid "goings and comings" from the support robots, and on the other hand the support robots will probably be able to handle several tasks at the same time, which will help to minimize the number of robots required for the pursuit mission (see section IV).

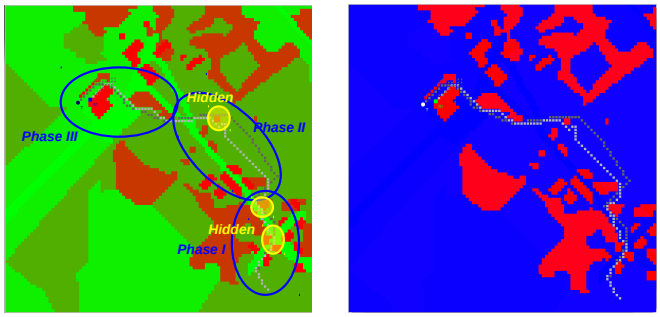
2) River-crossing situation: Figure 10 shows a situation where the target and the robot do not have the same traversability map, with an advantage for the former – imagine for example that the target is waterproof whereas the robot is not. While the target crosses the river, the robot has to take the bridge instead of directly following the target (phase I), which will probably leads to a temporary occlusion (phase II), until the robot reaches the target again (phase III).



(a) Robot traversability (b) Target traversab. (c) Robot visibility

Fig. 10. Pursuit in a river-crossing situation: (a) traversability for the pursuer (AAV), the river is the at the bottom of the terrain, a bridge in on the left; (b) traversability for the target (ground target); (c) visibility map. Difficulties arise when the target crosses the river, as the pursuer cannot follow it.

Figure 9-b displays the number of dangers and the temporal horizon evolution over the pursuit. One can again distinguish three phases: at the beginning there are risks that the target escapes (by breaking some visibility constraints); then as the target trajectory is not optimal the risks disappear, even if the target hides behind the building; at the end the pursuer catches up the target again. As the surrounding environment forms a dead-end, which is easy to handle, the temporal horizon reaches high values (over sixty time steps).



(a) Robot and target traversability (b) Robot visibility

Fig. 11. Pursuit in a realistic environment: (a) the pursuer (AGV) and ground target traversability; (b) the pursuer visibility. Difficulties arise in cluttered areas (phases I and III).

3) Experimental field situation: The third example is a realistic environment for ground target and pursuer, the vehicles evolve in both cluttered and rather clear areas (figure 11).

As figure 9-c shows, dangers exist near and within the cluttered areas, whereas in less difficult areas the greedy strategy performs well enough to prevent risks of failure (no need for support). In cluttered areas, the greedy strategy performs quite well too. It cannot prevent the target to escape with an optimal strategy, but is able to follow quite easily a not so adversarial target.

4) Real-Time performances: The previous sections states that considering a 9-connexity grid, the upper bound for the complexity in the graph construction is $O(h^3)$ in the worst case. It actually appears that the complexity is rather near quadratic (figure 12). This allows real-time computation, considering a value of 1 or 2 seconds for the time steps (figure 13), while keeping a reasonably acceptable temporal horizon (above 15). As a matter of fact, the temporal horizon is constantly adjusted to fit the computation time with the real-time constraint – that is, expand at least the graph of one time step in no longer than one time step.

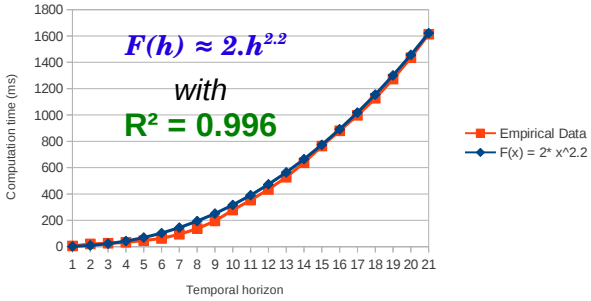


Fig. 12. Typical evolution of the computation time with the temporal horizon. Polynomial regression gives a near quadratic complexity (here : $F(h) \approx 2 * h^{2.2}$ with a very confident value for the coefficient of determination R^2).

Note on figure 13 that at few times the computation time exceeds the $1sec$ cycle criteria. This is actually because the system currently computes the states not one by one, but one horizon step at a time. As the prediction of the computational cost is not very precise, sometimes the system asks for one more horizon step and exceeds the time constraint. This is always balanced by the next computation cycles, which are very low. This would be solved by a slight adjustment of the implementation to compute the states one by one, improving both the respect of the time constraints and the overall performance (since more states should be computed).

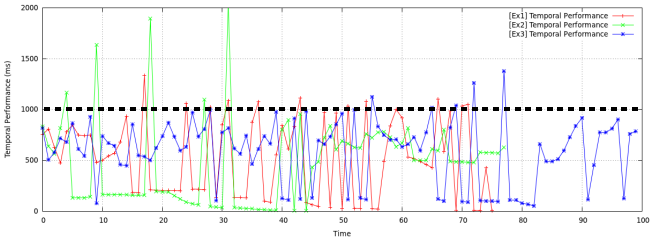


Fig. 13. Computation cycle at each time step for the tree previous examples. The dashed line represents the "one second" limit.

IV. Supplying team support

While a pursuer performs the following task, it predicts all its potential failures until a temporal horizon $\tau + h$, τ being the current time. These predictions straightforwardly generate a set a support tasks, defined as "watch place p at time θ", with p a cell of the spatial grid and $\tau \leq \theta \leq \tau + h$. By construction, the pursuer which issues these tasks cannot handle them. But it *manages* them: it allocates them to other robots according to their availability, capacities, priorities and the global objective of minimizing the number of required robots. Note that communication issues are currently not considered – they mainly

lead to situational awareness inconsistencies between robots, and may also lead the pursuer to have no (or not enough) support.

The set of support tasks is updated at each time step. Depending on the target motions, new tasks may occur and previous ones may become deprecated. The pursuer broadcasts the updates to the surrounding robots, which in turn update their current tasks list and evaluate if they could be of any help for the non-allocated tasks: this is a classical multiple task assignment problem.

Task allocation problems have given rise to a large amount of work, they raise combinatorial issues and finding optimal solution is still very difficult for non-trivial sized problem. Nevertheless many solutions have been proposed, for example using stochastic methods [15]. Auction-based approaches have been largely studied (see Dias et al. [6] for a survey) and have been shown to efficiently produce acceptable solutions. Choi et al. [4] presents decentralized algorithms which lead to good solution with some guaranties for optimum. The authors also highlight the combinatorial issue raised by the construction of the bundles of tasks (how to choose which subset of tasks will an agent handle) and the conflicts that result from the iterative construction of those bundles.

We have chosen to implement an auction process, as such approaches have shown good results and can handle inconsistencies within the situational awareness of agents. But in our case, costs are quite different from usual: as we aim to minimize the number of required robots, performing one or several support tasks has the same cost for a given robot, whereas reward is linear with the number of handled tasks. The auction process is one-turn, with multiple bids allowed: after the update of the task lists, each available robot computes several bundles of tasks it can handle together, and propose them to the auctioneer (the pursuer robot). Then the auctioneer combines those bundles and allocates subset of tasks (either a bundle or sub-part of a bundle) to the robots, trying to minimize the number of robots involved.

The support robots compute the bundles of tasks they can handle according to their capacities, the other tasks they already have, and their goals and priorities. As they cannot be exhaustive (n tasks lead to $\sum_{p=1}^{n} \binom{n}{p} = 2^n - 1$ possible bundles), they only compute the biggest bundles in an iterative way. We also want each task to be part of at least one bundle (if the robot is indeed able to perform the task). The results is an overlapping partition of the task set. Reminding the results of part III, the generated support tasks are spatially and temporally gathered, which helps to keep a small number of task bundles (usual values are 0 to 3). Tasks are also independent from each other (except for the temporal constraints) and thus can be handled separately.

Once these bundles are computed, the auctioneer gathers them and allocate tasks. It optimally combines the different bundles according to the following criteria :

(i) all the support task must be handled if possible
(ii) the number of required robots must be minimized.

This directly comes from the objective function (1). Finding the optimal combination of bundles could be expensive: considering N_t the average number of bundle for each robot and N_r the number of bidding robots, there are $O((N_t+1)^{N_r})$ possible combinations[3]. Typical maximum values are $N_t = 3$ and $N_r = 4$, which leads to $4^4 = 256$ possible combinations. This is still easy and quick to compute for the auctioneer using Boolean representation and bit-to-bit computation. Finally, combinations are ranked with the number of handled tasks (max) and the number of support robot involved (min), including the previously required robots for non-deprecated former tasks. Then the auctioneer sends the support robots the support tasks they have been allocated.

Note that the heavy computational load of the auction phase is deported on the bundles construction, sparing the auctioneer which is also the pursuer. It can thus spend more time computing the pursuit graph and expending the temporal horizon, which is the heart of the system. For the same reason we also introduce a temporal shift (of one time-step) between the time tasks are issued and the time associated bids are computed. Doing so allows support robots to compute bundles while the pursuer computes the next pursuit graph, avoiding the robots to await for the others. This shift is negligible compared to the expected horizon size ($1 \ll h$).

Work is still in progress for this part, but first results are promising and show that a minimum number of robots are involved, while still respecting our real-time constraints.

V. Discussion

We have presented a new approach for realistic target tracking problems. Keeping in mind that a team of robots often have several distinct objectives, we provide a new pursuer-centered cooperation which minimizes the number of required robots at any time, thus satisfying an economy of means principle. The approach is based on a single pursuer which calls for team support only when required, by the evaluation of future failures it can not handle alone. We exploit realistic multi-layered 3D models, which allow to consider different capacities (such as being waterproof or not) for both targets and robots. Kolling et al. [9] recently introduced 2.5D visibility in pursuit-evasion, but to the best of our knowledge it is the first time realistic multi-layered models as ours are used for such problems.

As our local following strategy performs quite well, only one robot is required most of the time, but team provide support when needed to prevent the target loss. The resulting system shows promising results in realistic simulations, and integration within a team composed of one AAV and two UGVs is under way. We intend to extend the models by considering communication constraints and failures: this is not a trivial issue, as communication constraints threaten the whole cooperation. Possible solutions are to extend the anticipation from the pursuer, and to integrate communication rendez-vous for the support robots.

[3] "$+1$" is for the empty bundle : each robot can either handle a bundle of tasks, or none. This is important to consider as we want to minimize the number of required robots.

Problems may also occur if the pursuer does not behave as planned (*i.e.* according to its pursuing strategy). However, in such situations the system is not totally failing, as re-planning and re-computing the risks on-the-go remain tractable. Besides, adjusting the spatial and temporal resolutions will lead to better anticipation and to more flexibility in the execution of the plans.

We also plan to integrate finer probabilistic motion models and thus to handle potential failures according to both their probability and their temporal proximity.

Finally, our longer term perspective is to merge our target tracking system within an overall surveillance scheme, aiming at having a whole team of robots performing under realistic conditions several tasks and achieving different goals in parallel.

ACKNOWLEDGMENTS

This research is partially funded by the DGA.

APPENDIX

We develop hereby the mathematical bounds presented section III. The mathematical proof stands for an environment free of any obstacle and in 9-connexity (using a 2D grid). Unexpectedly this is an upper bound for the complexity, as obstacles, viewlength and lower-connexity lead to motion restriction, and thus reduce the number of possible spatial states.

Under such hypotheses, we have, for each stage s in the graph obtained after spatial redundancies treatment, at most n_s nodes with $n_s \leq |Acc\,(target, s)| * |Acc\,(robot, s)|$ where $Acc\,(i, s) = A_{i,s}$ is the set of all accessible positions for i in time s. The total size N of the graph is under $\sum_{s=1}^{h} |A_{t,s}| * |A_{r,s}|$. Besides, in 9-connexity we have $A_{i,s} \leq 1 + \sum_{\sigma=1}^{s} 8\sigma = (2s+1)^2$, therefore we have $N \leq \sum_{s=1}^{h} (2s+1)^4 = O(h^5)$. Here we consider that the pursuer and the target have the same velocity: this does not restrict the generality of our proof, as different velocity will only change the resulting complexity ($|A_{t,s}| \neq |A_{r,s}|$), not the reasoning.

Adding temporal loops and iterative construction for the graph, one only develop the new nodes at each stage, referred to as the front nodes. Under the same hypothesis, we now have a number of front nodes N_f:

$$\begin{aligned} N_f &\leq (A_{t,h} - A_{t,h-1}) * A_{r,h-1} + (A_{r,h} - A_{r,h-1}) * A_{t,h-1} \\ &\leq (8h * O(h^2) + 8h * O(h^2)) \\ N_f &\leq O(h^3) \end{aligned}$$

REFERENCES

[1] T. Bandyopadhyay, Yuanping Li, M.H. Ang, and D. Hsu. A greedy strategy for tracking a locally predictable target among obstacles. In *Proceedings. ICRA 2006. IEEE International Conference on*, pages 2342–2347, may 2006.

[2] S. Bhattacharya and S. Hutchinson. On the Existence of Nash Equilibrium for a Two Player Pursuit-Evasion Game with Visibility Constraints. In *Algorithmic Foundation of Robotics VIII*, volume 57 of *Springer Tracts in Advanced Robotics*, pages 251–265. 2009.

[3] S. Bhattacharya, S. Candido, and S. Hutchinson. Motion strategies for surveillance. In *Proceedings of Robotics: Science and Systems*, 2007.

[4] Han-lim HL Choi, Luc Brunet, and Jonathan P How. Consensus-based decentralized auctions for robust task allocation. *Robotics, IEEE Transactions on*, 25(4):912–926, 2009.

[5] T. Chung, G. Hollinger, and V. Isler. Search and pursuit-evasion in mobile robotics. *Autonomous Robots*, 31:299–316, 2011.

[6] M.B. Dias, R. Zlot, N. Kalra, and A. Stentz. Market-based multirobot coordination: A survey and analysis. *Proceedings of the IEEE*, 94(7):1257 –1270, july 2006.

[7] B. Jung and G. S. Sukhatme. Tracking Targets Using Multiple Robots: The Effect of Environment Occlusion. *Autonomous Robots*, 13:191–205, 2002.

[8] A. Kolling and S. Carpin. Cooperative Observation of Multiple Moving Targets: an algorithm and its formalization. *Int. J. Rob. Res.*, 26:935–953, September 2007.

[9] A. Kolling, A. Kleiner, M. Lewis, and K. Sycara. Pursuit-evasion in 2.5D based on team-visibility. In *IROS 2010, IEEE/RSJ International Conference on*, pages 4610 –4616, oct. 2010.

[10] R. Mottaghi and R. Vaughan. An integrated particle filter and potential field method applied to cooperative multi-robot target tracking. *Autonomous Robots*, 23:19–35, 2007.

[11] R. Murrieta-Cid, R. Monroy, S. Hutchinson, and J.-P. Laumond. A Complexity result for the pursuit-evasion game of maintaining visibility of a moving evader. In *IEEE International Conference on Robotics and Automation*.

[12] R. Murrieta-Cid, B. Tovar, and S. Hutchinson. A Sampling-Based Motion Planning Approach to Maintain Visibility of Unpredictable Targets. *Autonomous Robots*, 19:285–300, 2005.

[13] R. Murrieta-Cid, R. Sarmiento, S. Bhattacharya, and S. Hutchinson. Surveillance strategies for a pursuer with finite sensor range. *International Journal on Robotics Research*, 2007.

[14] D. Portugal and R. Rocha. A Survey on Multi-robot Patrolling Algorithms. In *Technological Innovation for Sustainability*, volume 349 of *IFIP Advances in Information and Communication Technology*, pages 139–146. Springer Boston, 2011.

[15] Emilio Frazzoli Sertac Karaman, Tal Shima. Task assignment for complex UAV operations using genetic algorithms. *AIAA Proceedings.[np]. 10- ...*, 2009.

[16] Jorge Urrutia. Art Gallery and Illumination Problems. In *Handbook of Computational Geometry*, pages 973–1027, 2000.

[17] Christopher Vo and Jyh-Ming Lien. Visibility-Based Strategies for Searching and Tracking Unpredictable Coherent Targets Among Known Obstacles. In *ICRA 2010 Workshop: Search and Pursuit/Evasion in the Physical World: Efficiency, Scalability, and Guarantees*, Anchorage, Alaska, May 2010.

[18] L. A. Zadeh and J. H. Eaton. Optimal pursuit strategies in discrete-state probabilistic systems. *Trans. ASME Ser. D, J. Basic Eng*, 1962.

[19] Ke Zhou and S.I. Roumeliotis. Multirobot Active Target Tracking With Combinations of Relative Observations. *Robotics, IEEE Transactions on*, 27(4):678 –695, aug. 2011.