Cooperative Observation of Multiple Moving Targets: an algorithm and its formalization

Andreas Kolling Stefano Carpin*

School of Engineering
University of California
Merced – USA

Abstract

This paper presents a distributed control algorithm for multi-target surveillance by multiple robots. Robots equipped with sensors and communication devices discover and track as many evasive targets as possible in an open region. The algorithm utilizes information from sensors, communication, and a mechanism to predict the minimum time before a robot loses a target. Workload is shared locally between robots using a greedy assignment of targets. Across long distances robots cooperate through explicit communication. The approach is coined Behavioral Cooperative Multi-robot Observation of Multiple Moving Targets. A formal representation of the proposed algorithm as well as proofs of performance guarantee are provided. Extensive simulations confirm the theoretical results in practice.

1 Introduction

A growing application domain for multi-robot systems is data-gathering across large areas and exploration of unknown and hazardous terrain. This is of high interest for surveillance applications of all sorts, be it the observation of wild life in a preservation area or the surveillance of activities in regions of crisis. Information from the inside of a radioactively contaminated plant, an earthquake area, or chemically-contaminated regions can be most useful and save lives. Mobile sensors may also be used for known environments, to supplement traditional surveillance systems and possibly reduce deployment and maintenance costs. In particular, when moving targets are involved, mobile sensors may be of advantage if they can adjust to changes appropriately and in a timely fashion. Automated surveillance encompasses a great variety of research areas such as pursuit evasion games, path planning, sensor placement, target tracking and trajectory prediction, image analysis, map building, cooperative, behavioral and distributed robotics. Within this scenario, we focus on the Cooperative Multi-robot Observation of Multiple Moving Targets (CMOMMT from now on), firstly formalized by Parker in [18]. In this paper we present B-CMOMMT, a control algorithm whose early version was presented in [13]. B-CMOMMT, short for Behavioral CMOMMT, is a distributed algorithm that schedules three different behaviors, namely follow, explore and help, on every robot, so that the team cooperates to keep as many targets as possible under observation. Our original work has been here enriched with a more sophisticated mechanism to predict the minimum time before a target is lost, thus improving its original performance. Moreover, we here illustrate a sound mathematical formalization of the algorithm, an aspect often exluded in behavior based robotics. The provided formal analysis shows that the algorithm is *stable*, according to a definition formalized later on, and guarantees a long term lower bound on its performance.

Section 2 shortly reviews the state of the art. The problem is formalized in section 3, and the improved B-CMOMMT is thoroughly presented in section 4. Section 5 illustrates its formalization, proves the stability theorem, and gives a lower bound on the long term performance. Following the theoretical investigations, extensive simulations are presented in section 6 to demonstrate capabilities and properties of the improved version of B-CMOMMT. Finally, in section 7 conclusions and new directions for further work are offered.

*corresponding author: scarpin@ucmerced.edu

2 Related work

The most basic version of the target observation problem is the art gallery problem, a well known problem in computational geometry. The task is to find locations for cameras in a polygonal environment such that any point in the polygon at any time can be observed with the least number of cameras possible. The problem is NP-hard in general, but special cases allow to compute solutions efficiently (see [24] for an overview of this area). Also of interest, in particular for seeking new targets, is the problem of finding the shortest route for a watchman to once see the entire polygon. For polygons with holes, even when they are restricted to be convex or orthogonal, this problem is also NP-hard, although efficient solutions exist for polygons without holes [2]. Apart from the variations for the art gallery problem, there are other exact algorithms for simple environments such as [25] in which the task is to find a schedule for a flashlight to move along the boundaries of a simple polygon to detect possible intruders. This approach is a good solution for mobile sensors on rails on the boundary of the environment. A similar problem was studied by LaValle et al. in a stream of papers dealing with increasingly restricting hypothesis [7][8][14][21]. These computational geometry originated methods often search data structures that are extracted from geometrical features of the environment to find possible intruders in an environment. A similar approach was presented in [8] and has also been recently extended to a limited field of view by Gerkey et al. [5].

In close proximity to control theory we find a variety of distributed control algorithms for simple cooperative tasks like [3][4][9], and [15]. In [3] Cortés et al. present an algorithm for a network of mobile agents to achieve an agreement over the location in the network. The communication topology is represented via proximity graphs that capture the information on which agents can communicate to each other and are closest to each other. While moving, agents try to maintain connectivity, i.e. to follow a restricted motion to avoid losing communication links. Bullo et al. also present a distributed approach to the problem of optimal sensor coverage with mobile sensors given some starting position and a cost function [4]. The main challenge is to redesign an algorithm for the distributed computation of the paths of the mobile sensors and proving its correctness, i.e. convergences to a centroidal Voronoi configuration. In [9] Jadbabaie et al. provide a distributed algorithm for orienting a group of agents into the same direction. There are several variations of the algorithm, with and without agents that act as leaders. In all forms the standard agents compute their new orientation according to the average orientation of all neighbors. This way a common orientation should be reached. A third vein of research deals directly with surveillance issues and employs distributed control algorithms with complex cooperation and some heuristic elements. In many cases the algorithms are part of a real system and their analytical rigor is less ambitious than in the previous papers. In [22] the main emphasis is on target identification and recognition, path planning and the resolution of conflicting missions. Interesting in this approach is the complex architecture which combines all the high level tasks. The two papers most closely related to our work are [10] and [18]. In [10] Sukhatme et al. investigate a region based approach for surveillance of complex environments. Cooperation aspects are utilized to control the distribution of robots across regions. A robot is migrating to another region when it detects that the ratio of targets to robots is more than a defined threshold. In this case the robot moves to the most urgent region, a measure depending on the distances to the regions and the number of robots and targets in the region. Each robot has to maintain these variables from the broadcast messages it receives. For the local following of targets within a region each robot tries to maximize the number of targets observed by computing the center of gravity of all known targets and positioning itself a certain distance apart, depending on the field of view of the sensor and the maximum distance across targets to the center of gravity. More recently, Jung and Sukhatme [11] extended their region based approach to accommodate non simple environments and various restrictions for sensors. They also provided extensive simulation results with different mobile platforms. The paper by Parker [18] was actually the starting point for our work. The proposed solution drives robots around according to local directional vectors computed on the fly. This approach will be better illustrated in section 4 when describing the B-CMOMMT algorithm. It is also worth to outline that Parker [19] also provided one of the few formal treatments of behavior based multi-robot systems. This aspect is along the same spirit of the analysis we provide in section 5, although the used formalism is different. In terms of assignment of behaviors to robots, the paper by Werger and Mataric [26] is most closely related to our work. Our approach shares their principle for the assignment of tasks to robots, i.e. that only the most capable of the available robots receives eligibility to execute a behavior. This is achieved by a cross-inhibition of the same behaviors on different robots and a local subsumption hierarchy that determines the active behavior for a particular

robot. Plenty of researchers from various areas have produced interesting results in the area of prediction of target movement. Knowing future locations of targets is of interest for collisions avoidance in path planning. Furthermore, sensor networks utilize predicted trajectories of targets for energy conservation by computing sleep and activation phases based on the predictions. In the field of estimation of target positions from sensor data some predictive elements are also found. For our prediction mechanism we shall present some alternatives from this research area in section 4, when discussing our proposed prediction schema.

3 CMOMMT formalization

CMOMMT has been first formalized by Parker in [18]. With minor variations, we here propose the same notation and terminology introduced therein. Let:

1. S: two dimensional, bounded obstacle free area of interest.
2. O: a set of n moving targets, $o_j, j = 1, 2, \ldots, n$. The position of o_j at time t shall be denoted by $\vec{o}_j(t)$. Targets cannot leave the area of interest, i.e. $\forall t\ \vec{o}_j(t) \in S$.
3. V: team of m robot vehicles, $v_i, i = 1, \ldots, m$. The position of v_i at time t shall be denoted by $\vec{v_i}(t)$. It is assumed that each robot carries sensors able to detect the targets.
4. $SC(v_i, t)$: the *sensor coverage* is the subset of S observable by robot v_i at time t[1]. This region varies as the robot v_i moves inside S, but its shape is supposed not to vary. The maximum sensing range of robot v_i, denoted as $r_s(v_i)$, is defined as follows:

$$r_s(v_i) = \max_{p \in SC(v_i,t)} \|\vec{v}_i - p\|_2$$

   where $\|\cdot\|_2$ indicates the Euclidean distance. In an homogeneous team, the common sensing range is indicated as r_s.
5. $B(t)$: $m \times n$ observation matrix such that $b_{ij} = 1$ if robot v_i is observing target o_j at time t, 0 otherwise. Robot v_i observes target o_j at time t if $\vec{o}_j(t) \in SC(v_i, t)$.

The goal is to develop an algorithm that maximizes the following metric for surveillance:

$$A = \sum_{t=1}^{t_e} \sum_{j=1}^{n} \frac{g(B(t), j)}{t_e}, \tag{1}$$

where $g(B(t), j)$ is 1 if there exists an i such that $b_{ij}(t) = 1$ and 0 otherwise. The time t_e is the execution time of the algorithm. Informally stated, the problem requires to maximize the average number of targets that are observed by at least one of the robots. Restricted overall sensor coverage is furthermore assumed. This can be expressed as:

$$\bigcup_{v_i \in V} SC(v_i, t) << S,$$

for any t. This prevents the use of static robot placements, since under this hypothesis an intelligent target can easily position itself in a point not covered by any of the robots. Moreover, we assume that the maximum speed of the targets is smaller than the maximum speed of the robots. If this is not the case, it would be easy for an intelligent target to always escape robots by just moving at maximum speed. Let us denote these two speeds as s_o and s_v respectively. Additionally, all robots share a common global reference system. Finally, we assume that robots have a communication mechanism, which allows them to send or receive messages in broadcast mode. A robot can receive any message sent by any other robot closer than the *communication range* r_c. Concerning sensing, differently from [18], we do not necessarily require that robots are equipped with omnidirectional sensors, nevertheless supposing that only targets within distance $r_s(v_i)$ from the robot can be detected by v_i. Most of our simulations and the formal part, however, still assume omnidirectional sensors. Indeed, an extension to a limited field of view would require changes in the computation of the local

[1] the sensor cover function could be equivalently written as $SC(\vec{v}_i(t))$

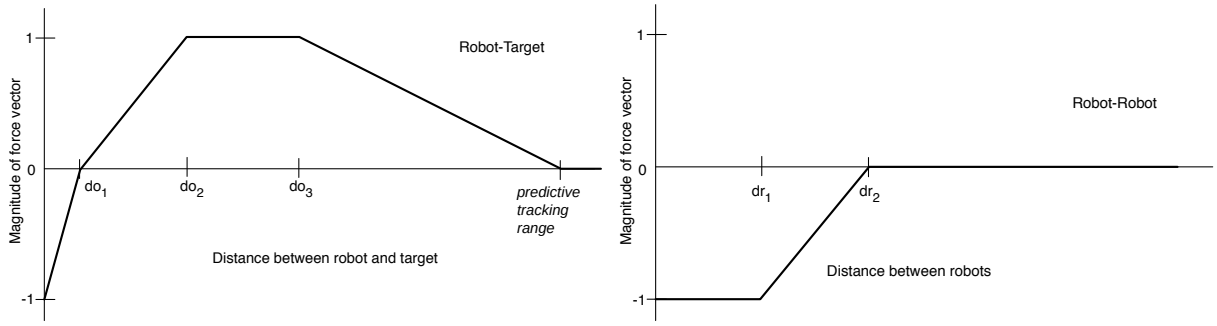

Figure 1: On the left: re-scaling factor for the vectors $f^t_{i,k}$ from robot to target. On the right: re-scaling factor for the vectors $f^r_{i,k}$ from robot to robot (from [18]).

force vector. The main components of B-CMOMMT, however, do not require much change when extended to a limited field of view and we believe that with some further work the assumption can be dropped entirely. Some experiments in [13] already test B-CMOMMT with varying field of view. For our approach we will adopt the same hypotheses stated in [18], i.e. that the sensing range r_s is smaller than the communication range r_c. All the above hypothesis, in particular with respect to communication and localization, are nowadays achievable both indoor and outdoor with off the shelf components.

4 B-CMOMMT

This section is divided into two parts. We first describe the general principle governing B-CMOMMT, as they were first presented in [13]. This serves as reference for the second part, where improvements are proposed and described.

4.1 The basic B-CMOMMT algorithm

A first algorithm to solve the CMOMMT problem is proposed by Parker in [18] and called A-CMOMMT. It is based on a local vector strategy which drives the robot by computing a desired direction vector using the vectors to targets and other robots. It incorporates the following principles: 1) stay close to targets that are not too far away; 2) stay away from other robots to cover more terrain and avoid collisions; 3) stay away from targets that get too close to avoid collisions; 4) have all surrounding robots and targets influence the movement of the robot. This is expressed by the following formula:

$$F(v_i, t) = \sum_{k=1}^{n} w_{ik} f^t_{ik} + \sum_{k=1}^{m} f^r_{ik}. \tag{2}$$

where $F(v_i, t)$ is the desired relative goal position for robot v_i at time t, f^t_{ik} is the rescaled vector from robot i to target k, and f^r_{ik} the rescaled vector from robot i to robot k with $f^r_{ii} = 0$ at time t. These two terms depend exclusively on the target-robot and robot-robot position. The precise profiles for the re-scaling factor of the distance vectors are given in figure 1. A total of 5 variables have to be set in order to define the numerics of the profiles. The *predictive tracking range* appearing in figure 1 is the range within which targets located by robots other than v_i or targets that left the sensing range influence v_i's motions. The location of targets beyond sensing range and closer than *predictive tracking range* is predicted linearly using the last known movement of the target. The w_{ik} factors account for cooperation between different robots. The idea is to share the workload between different robots. If target o_j is currently being observed by robot v_k, then robot v_i should be less attracted to observe o_j. This turns out in a choice of w_{ij} smaller than 1. Extensive simulation results presented in [18] illustrate that the introduction of the w_{ik} factor yields an approach that outperforms the local force method, i.e. without w_{ik}, in many scenarios.

In our B-CMOMMT approach we view CMOMMT in the framework of a decision process. This view is justified by the fact that we are concerned with a decoupled system, in which each robot has access to local

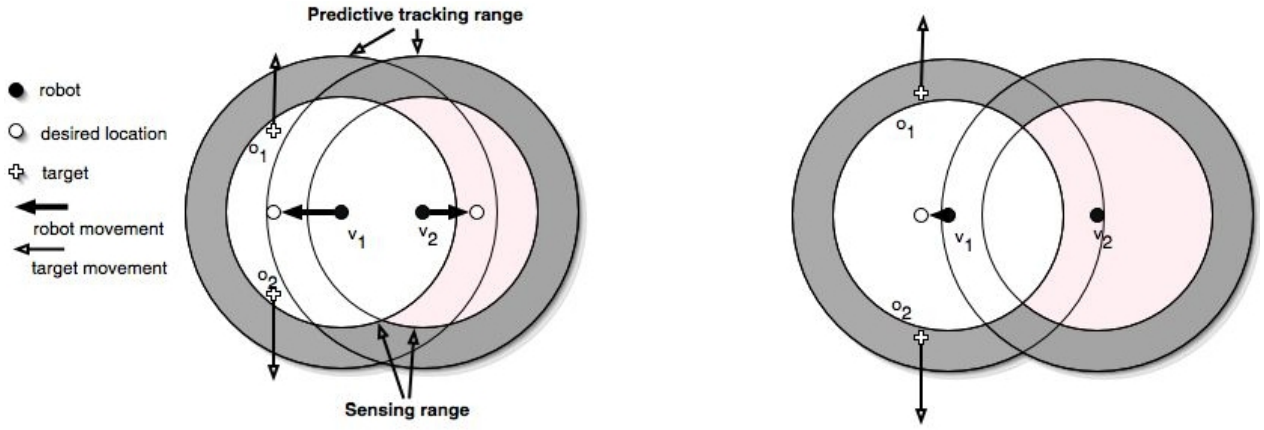

Figure 2: A problematic situation for A-CMOMMT. One robot following two targets with another robot close by (figure taken from [13]).

information via sensors and via other robots that communicate with it. Provided with this local information, a robot has to decide on the impact of the targets and robots on its movement. The starting point was that we wanted to overcome some problematic situations that may arise in the A-CMOMMT approach. One of these situation was when one robot followed two targets that move in opposite direction, eventually losing both. The robot moves into the center of mass of the two targets and when the targets escape, they escape in the same time step and at very similar distances. Only if one target got lost earlier the robot would be able to catch the other target as only then it can ignore the force vector of the second target. The loss of both targets may happen even when another robot is close by. The other robot is repelled from the first robot due to the force vector from that robot and has no chance of recapturing any of the targets. Intuitively the second robot should have attempted to capture one of the two targets instead of wandering around and attempt to discover an unknown target. A sketch of this situation can be seen in figure 2. A more detailed discussion can be found in [12] and [13]. B-CMOMMT operates under similar assumptions as in the problem definition of CMOMMT, with the difference that it accommodates for varying restrictions of the sensors. It uses the same profiles for the re-scaling of local directional vectors used in A-CMOMMT and can be seen as a high level control to set the weights in the formula for the weighted vectors. An important difference is that we also have weights for robots, i.e. to rescale the f^r_{ik} contributions. B-CMOMMT realizes a simple behavioral architecture. A robot can be in one of three modes and broadcast help calls to indicate future target loss. The modes are:

1. **Follow**: in the mode "Follow" a robot is following one or more targets that it observes.
2. **Help**: In "Help" mode, when no targets are in sight, it drives towards a robot within communication range that requests help.
3. **Explore**: in "Explore", when no robot requests help and no targets are detected, it seeks new targets.

A help call is broadcasted when a target loss of a currently observed target is predicted. At an earlier stage of this research [13], we used a simple heuristic for the prediction of target loss, namely to call for help when a target enters the predictive tracking area. In the second part of this section we present significant improvements to the prediction system. The behavioral architecture is supported by a *tagging* system, which operates under the principle that one target should influence the movement of at most one robot at a particular time. An overview of the modes is given in figure 3. The proper tagging of targets requires the additional hypothesis that targets are distinguishable, a prerequisite not necessarily present in the CMOMMT problem formulation, but made also in [18]. To achieve target identification one can use the global reference frame and identify targets by their position. In case only insufficient and unreliable information on target positions is available, one should use existing approaches such as in [22]. The tagging evolves as follows. Firstly, every robot tags all targets within sensing range. For each tag the robot communicates the distance of the

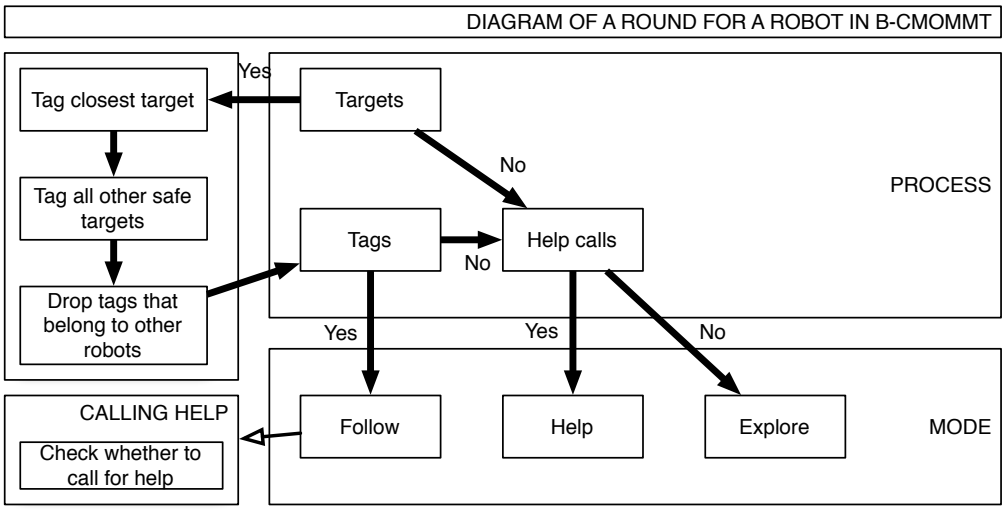

Figure 3: Overview of the modes in B-CMOMMT.

tag to the other robots within communication range. If two robots tag the same target, the one at larger distance drops the tag once it realizes this fact through the broadcast messages. With this approach, it suffices to have the communication range twice as large as the sensing range to avoid duplicate tags on the same target. The rationale behind tagging only within the sensing area is that a robot should be able to still follow any tagged target if it was to decide to follow such target exclusively. While targets are tagged by a robot they will all influence the robot's desired direction for the movement in the next time step with full weights, so that the robot attempts to move towards the center of mass of these targets. Resources are efficiently used because there will never be multiple robots following the same targets in situations where a target loss is not foreseen. After the tagging is done the mode of the robot can easily be determined. If a robot has a tag on at least one target it switches into "Follow" mode. If it has no tags and it receives a help call from another robot it switches into "Help" mode. If no tags and no help calls are available it switches to "Explore" mode. Extensive experimental results presented in [13] indicate that B-CMOMMT performs better than A-CMOMMT in scenarios with restricted resources, i.e. target to robot ratios of 3/1 and 2/1.

4.2 Improvements to the B-CMOMMT algorithm

Among the contributions of this paper is a direct improvement of B-CMOMMT by replacing its heuristic based target loss prediction with more advanced methods. A further improvement consists in the definition of more elaborate help calls in order to decrease the likelihood that target loss occurs. This subsection details about these two refinements.

4.2.1 Improved target loss prediction

The following discussion is robot-centric, i.e. it describes the process from a single robot point of view. It is assumed that the same procedure is carried out by every $v_i \in V$. Being able to determine the time when a target will escape observation is inarguably an important aspect of any target surveillance algorithm. Even more so when mobile and cooperating sensors are present, since a predicted target loss can be avoided if other mobile sensors are available. In formal terms the ability to predict target loss is the ability to compute the probability $p_j(t)$ that target o_j will escape observation within time t. In general p_j is a non decreasing function. To be able to talk about a particular time of loss t_l of a given target o_j we define it as:

$$t_l = \min\{\ t \mid p_j(t) \geq \delta\}, \tag{3}$$

where $0 < \delta \leq 1$ defines from which probability onwards we consider target loss to occur. To estimate t_l or at least a lower bound t_l^* we have to determine p_j. For practical implementations t_l^* suffices, since actions to

recapture the target will just be initiated earlier. Let us consider robot v_i and define the following:

Definition 1 (Tag set) *The* tag set $T(v_i, t)$ *is the set of targets tagged by robot* v_i *at time* t.

Primarily we are interested in the time of loss for the first target to get lost which we shall denote by t_f. According to the previous arguments, we have to determine p_j for each target in $T(v_i, t)$ and pick the smallest. A simple reformulation shows that we can avoid this. Using equation 3 for t_f we get:

$$t_f = \min\{\ t \mid \exists\ o_j \in T(v_i, t), s.t.\ p_j(t) \geq \delta\}. \tag{4}$$

Since a robot might have multiple targets, computing the function for each target may be costly. To avoid computing each p_j we consider a function

$$p'_{v_i}(t) = \max_{o_j \in T(v_i,t)} (p_j(t)), \tag{5}$$

This function p'_{v_i} combines the information from all p_j from a tag set $T(v_i, t)$. Figures 4 and 5 illustrate this simple modification. An algorithm to compute $p'_{v_i}(t)$ may e.g. utilize certain relations such as $p_1(t) < p_2(t)$ to avoid the computation of certain p_j and more advanced learning methods could directly extract information on $p'_{v_i}(t)$.

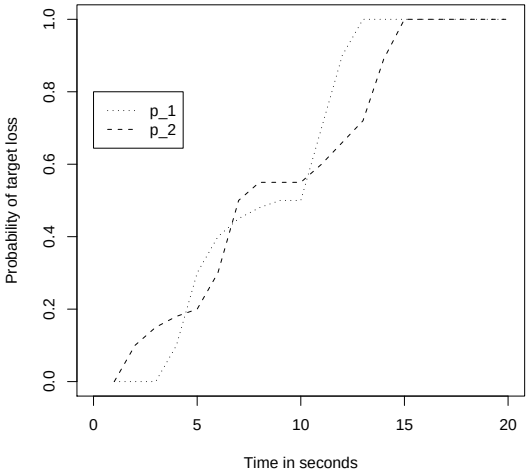

Figure 4: The graph shows a possible estimate for the two functions p_1 and p_2 for targets o_1 and o_2.

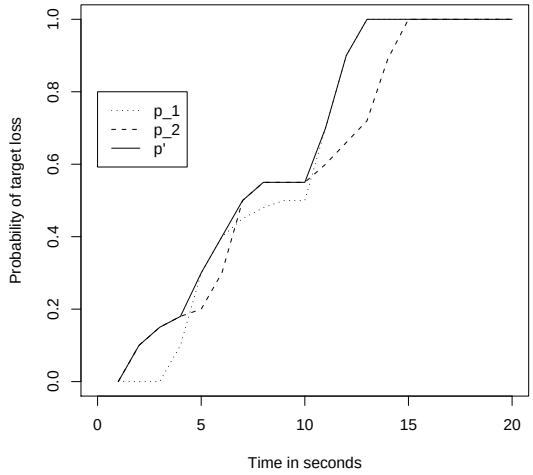

Figure 5: The graph shows the same as figure 4 with p'_{v_1} added, assuming the tag set of v_1 has only o_1 and o_2 in it.

Using p'_{v_i} we now get:

$$t_f = \min\{\ t \mid p'_{v_i}(t) \geq \delta\}. \tag{6}$$

Now, we can determine a lower bound on t_f, denoted by t^*_f, by only determining p'_{v_i}. Depending on the prediction mechanism employed it may be easier to use p'_{v_i} or all p_js. The prediction of target loss, i.e. an estimation of p_j or p'_{v_i} or parts of these functions, has two basic approaches. The first approach is *direct* and involves movement prediction for individual targets. The second is *indirect* and uses previously experienced target loss to estimate future target loss. Direct target loss prediction is most suitable for determining p_j while indirect target loss prediction is more suitable for determining p'_{v_i}. We will illustrate both mechanisms.

Direct target loss prediction. At the basis of direct target loss prediction is a method to compute a continued target trajectory. This prediction can range from linear continuation, to probabilistic estimation of likely trajectories depending on the knowledge about the target movement. Here we show how one can

utilize the trajectory prediction to efficiently calculate the time of target loss. As first example let us consider the case with one robot, two targets with evasive target movement, holonomic drives and no inertia. Evasive target movement means that targets can detect all robots sensing them and move away from them using a local force vector approach. We assume that o_s, sensing range of the targets, is equal to r_s. Evasive target movement is non-linear due to the interaction between the robots and targets. The equations for the robot and targets movements define two dynamically changing vector fields, one for the robot and one for targets. We want to simulate the situation when the robot has the two targets within sensor coverage until one of them is lost. For this it suffices to consider the case when the robot and targets are influencing each other and ignore the case when they lose each other out of the sensor coverage since this denotes the end of the simulation. Let $\vec{x}$ represent the current state of the considered subsystem, i.e.

$$\vec{x} = \begin{pmatrix} \vec{v}_1 \\ \vec{o}_1 \\ \vec{o}_2 \end{pmatrix}.$$

We have an autonomous dynamical system with the following equations:

$$\frac{\delta \vec{v}_1}{\delta t} = \frac{\sum_{j=1,2} f^t_{1,j}}{\| \sum_{j=1,2} f^t_{1,j} \|_2} \cdot \min(\| \sum_{j=1,2} f^t_{1,j} \|_2, s_v) \tag{7}$$

$$\frac{\delta \vec{o}_1}{\delta t} = \frac{\vec{o}_1 - \vec{v}_1}{\|\vec{o}_1 - \vec{v}_1\|} \cdot s_o \qquad \frac{\delta \vec{o}_2}{\delta t} = \frac{\vec{o}_2 - \vec{v}_1}{\|\vec{o}_2 - \vec{v}_1\|} \cdot s_o \tag{8}$$

$$\vec{x}' = F(\vec{x}) = \begin{pmatrix} \frac{\delta \vec{v}_1}{\delta t} \\ \frac{\delta \vec{o}_1}{\delta t} \\ \frac{\delta \vec{o}_2}{\delta t} \end{pmatrix} \tag{9}$$

where s_o is the maximum target speed, while s_v is the maximum robot speed. The dimensionality of this system increases with every additional target since there are two spatial dimensions per robot and target. Due to the necessary normalization of the vectors we have a non-linear system. Let $|T(v_i, t_0)| = k$ and let $\vec{m}_i$ be the observed movement vector of the target $o_i \in T(v_i, t_0)$. Assuming that $T(v_i, t_o)$ does not change, the time of loss is the t solving the following equation:

$$r_s(v_i) = \left\| \frac{\sum_{o_i \in T(v_i,t)} (\vec{o}_i + t \cdot \vec{m}_i)}{k} - \vec{o}_j \right\|_2, \text{for any } o_j \in T(v_i, t_0), \tag{10}$$

i.e. when the center of gravity of all tagged targets is further away from one of these targets than the sensing range of the robot, then the robot will lose such target. Note that we are assuming that the robot is capable of moving into the center of gravity sufficiently fast. This assumption is reasonable as for a constant set $T(v_i, t)$ the center of gravity can at most move as fast as the targets maximum speed. Equation 10 can be solved analytically. The resulting time of loss is an approximation to the actual time of loss which may not even be a lower bound but higher than the actual time of loss. Clearly, this method is very simplistic but also practical.

Indirect target loss prediction. Indirect target loss prediction is a more general approach that may be used in different environments with varying targets without much adjustment. The goal is to capture patterns in previous target movements with learning, and thereby predict future target loss. The main challenge is to find an encoding that captures most of the dynamics of the situations and allows sufficiently accurate predictions. Generally, about almost everything can influence the time a target will be lost out of observation. The configuration and differential constraints on robots and targets, the number of robots and targets and the environment play all a role. To illustrate how B-CMOMMT can integrate the results of a prediction, we are going to present the most basic approach which incorporates the essential information, i.e. the relational movement of the tags with respect to the robot. This relational movement is encoded by the distances between the robot and all its tagged targets, and by the target to target distances. The rationale is the following: the closer all tags are to each other and to the robot the less likely it is for a target loss to occur soon. This approach works rather well in practice as we shall see in section 6, even though dynamical constraints are completely ignored.

Regardless of what information is encoded and used for the learning, the robots need a method for acquiring it. For our implementation of B-CMOMMT we collected training data at a central instance after several test runs. We use a simple neural network to predict the time at which a target will be lost. The prediction mechanism is implemented for two and three targets. The two target case is explained in more detail. The construction of the three target case is analogue. We chose the target movement to be evasive, i.e. targets are repulsed by all nearby robots as in [18]. This type of target movement is sufficiently complicated while on the other hand still exhibiting a distinct pattern as sketched in figure 6. As basis for the training

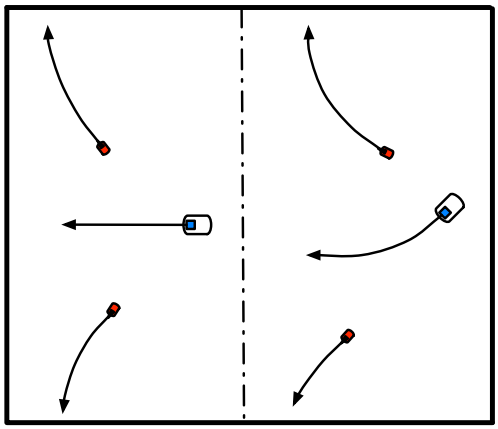

Figure 6: Two scenes with a robot following two targets that move evasively

data we used log files from 50 runs each with 10 robots, 10 targets and a duration of 4 minutes. The log files are parsed for a situation in which one robot is following two tagged targets until one of the targets escapes the observation without any intervention of other robots and targets. Such a situation occurred 253 times in the logged test runs. Each situation contains multiple observations starting from the time when the robot started to follow two targets until it lost one. For each observation we computed the time to the target loss and the relative distances of the tags to each other and to the robot. Out of all observations a total of almost 11000 training instances were created this way. Since we are considering the case with two tags, one training instance consists of the distance of the robot and the two tags rescaled onto $[0, 1]$ by division through r_s and the distance of the two tags to each other, also rescaled to $[0, 1]$ by dividing it by $2 \cdot r_s$. The teacher signal for a training instance is the time of loss also rescaled to $[0, 1]$ by division by 20. The number 20 was chosen since we intended to predict up to 20 seconds in advance. Situations with a tag loss later than 20 seconds are not considered.

For the set up of the neural network, a network with one output neuron, three input neurons and one intermediate layer with 6 neurons was chosen. It produced a training MSE (Medium Square Error) of 0.0045632105. The learning seemed to have grasped the very simple pattern effectively. No mechanism was used to detect outliers that are clearly visible in the data. Figure 7 offers a pictorial representation of the training data. The trained network was then integrated into the client programs for B-CMOMMT and used to predict the time of loss whenever two targets were observed. The same was done for the three target scenario. For more than three targets the direct linear prediction formerly introduced was used.

4.2.2 Modified help calls

In the initial B-CMOMMT implementation, target loss was identified when the target left the sensing range and entered the linear predictive tracker. In such cases target loss is almost inevitable, unless by chance another robot is close by and able to help in due time. Furthermore, there was no mechanism to provide information on the positions or number of the targets to be lost. These shortcomings have been leveraged by the extension we describe in the following. A help call in the new version of B-CMOMMT is now associated to a target, i.e. one robot can issue multiple help calls and at most one for each target it has tagged. The prediction mechanism allows to add the information on the time of loss to the help call. The position of the

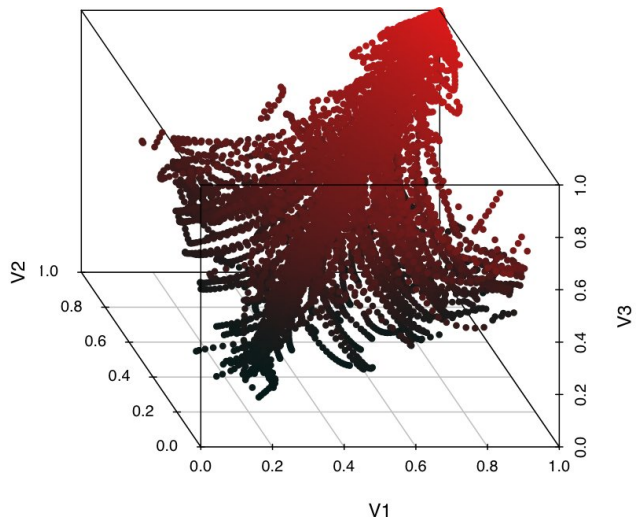

Figure 7: A *four dimensional* plot of the predictions produced by the neural network. Dimensions $V1$ and $V2$ are the distances between robot and tag 1 and 2 respectively, while $V3$ is the distance between tags (distances are normalized to the range 0-1 for displaying purposes only). The fourth dimension is represented by the color intensity. A black point represents a time until target loss of 20 seconds, while a bright point is an immediate target loss.

target is also added. A help call then contains the target identification, its position and expected time of loss. This modification will also allow us in section 5 to prove the stability results. Note that we are assuming a shared clock in order to interpret the time of loss. A shared clock can be maintained in a distributed network as shown in [16]. To summarize, the new help call system is designed as follows: first the robot identifies its tags and predicts target loss. Then for each target that is predicted to be lost it initiates a help call indicating the time of loss and current position of the target. This, however, only applies when the target loss prediction is able to predict the time of loss for each target. In some cases we may only have a mechanism that predicts the time of loss for the first target to get lost. In this case we should use the following: first the robot identifies its tags and predicts the time until the first target is lost. Then for all targets, except the closest, it initiates a help call indicating the time of loss and the currently sensed position of the target. This strategy is more conservative since any target, apart the closest, could be the target that is first lost. Picking the closest as the safe target is a heuristic. In case a help request reaches more than one helping robot, available helpers have to agree on who will answer the request. For the assignment of help calls one is faced with a similar problem as with the assignment of tags to targets by multiple robots. The difference, however, is that help calls control a long distance distribution of robots, i.e. within r_c, while the formerly described tagging procedure is relevant for a local distribution, because a tag can be at most r_s away. Bearing this in mind, the two assignment mechanisms shall resemble this distinction. This assignment problem is closely related to the one studied in [23] by Schenato et al. The task analyzed therein is to assign k robots to capture k targets. For this purpose, a measure called "time to capture", T_r, is introduced. It is defined as the time a robot needs to move into exactly the same configuration where the target is. The target trajectory is continued linearly while the robot can move according to allowed control inputs. Since the trajectory of the target is linear and the locations of robots and targets are known, as well as the allowed control inputs, T_r can be estimated by computing a control input that minimizes it. In our case we are looking for an assignment of help calls that does not necessarily minimize the time to capture for all targets but that rather ensures that all targets are captured before they are lost out of the sensor coverage. For these purposes a modified definition for T_r is needed since we are considering a target caught when it is within sensor coverage. Let k be the number of robots in help and explore mode and h be the number of help calls. $T_r(i,j)$, the time needed by robot v_i to sense target o_j, is defined as follows:

$$T_r(i,j) = \min_t\{t \mid \vec{o}_j(t_0+t) \in SC(v_i, t_0+t)\}, \tag{11}$$

where $1 \leq i \leq k, 1 \leq j \leq h$. For the computation of T_r one could consider using the methods and results from [23], where this problem is formulated as an instance of the *linear bottleneck problem*. However, such an approach does not fit into the B-CMOMMT paradigm, even though it gives an optimal assignment of help calls. Most importantly, it requires a central instance for the computation, which is not available in our set up. While it may be interesting for certain scenarios, we are going to focus on a distributed approach for the assignment of help calls that is easier to implement and also usable for robot teams with limited communication range. The distributed help call assignment proceeds as follows. All help calls that are issued but not assigned to any robot are considered unanswered help calls. All exploring robots read the communication channels for unanswered help calls within their communication range. If such a robot finds an unanswered help call for which the expected time of the target loss is smaller than the time needed to get the target into sensor coverage it answers it by tagging it and broadcasting this information. Only the closest such help call is chosen. The help call is now considered answered. Once a robot chooses a help call it commits to it, i.e. it does not check for other help calls as long as its tagged help call is still active and not overwritten by another robot. It continues exploring the environment as long as it can still capture the target. Once the difference between the time to capture the target and the time to loss exceeds a defined threshold h_m the robot stops exploring and starts to move to bring the target into the area covered by its sensor. More precisely, when $T_r > t_l - h_m$ for a small $h_m > 0$, then the robot moves towards the target. The threshold h_m is practical since one would like the robot to answer the help call securely, i.e. not precisely in the last possible moment, but with some additional time to accommodate for errors. In section 7 we will elaborate more on this topic. Once in help mode and recapturing the target, a robot tags only targets that do not have a tagged help call or no help call and no following robot. Targets with a tag and no help call and targets with a tagged help call are considered to be safely under observation. This mechanism ensures that a help call once tagged will be answered. Furthermore, it excludes extensive switching of help calls between robots, allowing the helping robot to continue with exploration first until the help call becomes urgent. However, switching of help calls with a tagging system is allowed in certain circumstances. Each robot checks the help calls and chooses the closest unanswered help call that it can successfully answer. If no such help call exists it checks the answered help calls and the distances of the answering robots to the help calls. It then picks the closest help call for which it is closer than the answering robot. This frees the formerly answering robot, which can continue exploring while the closer robot takes over the help call. The pseudo-code in algorithm 1 on page 12 shows how the help call system can work in a distributed variant using a mechanism that can only predict the time of loss of the first target, i.e. the version that makes the least assumption about the capabilities of the system which is also the one used for our simulations.

Figure 1 defines the desired following distance with the profile of the re-scaling of the directional vectors. Assuming a robot follows one target, then it will after some time follow it at distance d_{o_1}. In a general approach, however, we would like to provide an upper bound for the following distance. We are assuming that in the best case the robot can be directly on the position of the target, i.e. right above it. This is possible in the case of aerial pursuit. The upper bound on the desired following distance ensures the proper working of B-CMOMMT. The upper bound is defined in relation to the number of robots sensed by robot v_i, let this number be r.

$$d_{o_1}(v_i) \leq \frac{r_s}{2 \cdot (r+1)} - \xi \tag{12}$$

The adjustment by ξ is used in the theoretical section 5 and accounts for the time a robot needs to turn and accelerate into any direction to its maximum speed.

5 Theoretical analysis

The formal representation of B-CMOMMT is based on a matrix representation. We start by providing formal definitions of the terms being used in the following. Most of the structures defined in this section model relationships that vary over time, due to the motion of robots and targets. For this reason they are defined as functions of time t.

Definition 2 (The environment S) *The environment S with the set of robots and targets is modeled by a weighted graph $E(t)$ defined as follows:*

Algorithm 1 Move_robot(i)

```
Mode ← Explore
loop
  TT , UH, AH ← ∅
  Read sensor data to discover targets
  for all targets sensed do
    Read the tag of the target
    Compute distance to target d_t
    if distance in tag > d_t OR robot id in tag == i then
      Add target to TT
      Broadcast tag for this target
    end if
  end for
  Predict t_l the time of loss for the first target
  for all targets in TT do
    if target is not the closest then
      broadcast help call for target
    end if
  end for
  if TT == ∅ then
    if Mode == Help then
      if time to capture < time to loss then
        Explore the environment
      else
        Capture the target of the help call
      end if
    else if Mode == Explore then
      for all received help calls do
        Compute time to capture target t_c for help call
        if Help call is answered then
          if t_c smaller than most recent answer to help call then
            Add help call with t_c to AH.
          end if
        else
          Add help call with t_c to UH
        end if
      end for
      if exists a closest help call in UH then
        Help call ← closest help call in UH
      else if exists a closest help call in AH then
        Help call ← closest help call in AH
      end if
      if Help call ≠ 0 then
        Broadcast answer to help call
        Mode ← Help
      else
        Mode ← Explore
        Explore the environment
      end if
    end if
  else
    Follow targets in TT
  end if
end loop
```

- $Vertices[E(t)] = \{n_1, n_2, \ldots, n_{m+n}\}$. *It will be implicitly assumed that* $n_i = v_i$ *for* $1 \leq i \leq m$, *and* $n_i = o_{i-m}$ *for* $i > m$
- $Edges[E(t)] = \{[n_i, n_j] : 1 \leq i \leq m,\ m < j \leq m+n,\ \vec{n}_j \in SC(n_i, t)\}$
- $Weights[E(t)] : w([n_i, n_j]) = \|\vec{n}_i - \vec{n}_j\|_2$

From this information on visibility and distances the assignment of tags is defined. We can conveniently regard the tags as a subset of the edges of the graph $E(t)$, i.e. a tag, as a target to robot assignment, is an edge selected from $Edges[E(t)]$. This concept is formalized by the next definition.

Definition 3 (Tags and tag set) *A* tag *is an edge* $c = [n_i, n_j] \in Edges[E(t)]$ *which satisfies the following criterion:* $\forall l \neq i : [n_l, n_j] \in Edges[E(t)] \Rightarrow w(n_i, n_j) < w(n_l, n_j)$. *The set of all such edges* c *is called the* tag set *and is indicated as* $T(t)$. *The tag set of a robot* n_i *at time* t *is indicated by* $T(v_i, t)$ *and defined as follows*

$$T(v_i, t) = \{[n_i, n_j] \ : \ [n_i, n_j] \in T(t)\}.$$

The reader should notice that the above definition redefines in different terms the *tag set* already introduced in definition 1.

Definition 4 (Tagging graph and adjacency matrix) *The* tagging graph $g(t)$ *is a subgraph of* $E(t)$, *defined as follows:*

$$Vertices[g(t)] = Vertices[E(t)]$$
$$Edges[g(t)] = T(t)$$

The adjacency matrix $\Phi_{g(t)}$ *is an* $m \times n$ *matrix i.e.*

$$\Phi_{g(t)}(i,j) = \begin{cases} 1 & \text{if } [v_i, o_j] \in T(t) \\ 0 & \text{otherwise.} \end{cases}$$

$\Phi_{g(t)}$ may be considered as a modified adjacency matrix of the graph $g(t)$. A full $(m+n) \times (m+n)$ adjacency matrix is not needed since no edges can exist between two nodes n_j and n_k with $k, j > m$. Next, we define the *distance vector* that measures the distances between the tagging robot and the tagged target.

Definition 5 *The* distance *vector* d *is an* n *dimensional vector defined as follow:*

$$d_j = \begin{cases} w([n_i, n_j]) & \text{if } [n_i, n_j] \in T \\ 0 & \text{otherwise} \end{cases}$$

The graph $g(t)$ contains all information needed for the following and exploring. If the help call assignment in the centralized and optimal version and a simple prediction mechanism as in section 4 are used, then $g(t)$ suffices to describe the entire system fully and it is the only data structure that needs to be maintained. For the distributed assignment of help calls a memory of the help call assignment of the previous round is necessary. A change in the tagging graph $g(t)$ has to be the result of a tag drop, discovery of a target or a tag exchange between two robots. The tag drop removes an edge in $g(t)$, the discovery adds one, and the tag change moves an edge within g and reduces its weight.

Time to capture. The computation of the time to capture can be done as described in section 4. Alternatively, as done in the simulations presented in section 6, to estimate the time to capture target o_j the following equation can be used:

$$T_r = \frac{d_j - r_s}{s_v - s_o} + \delta \tag{13}$$

where δ denotes the time the robot needs to turn into the direction of the target and accelerate, and d_j is the distance between target o_j on which the help call has been issued, and the robot that has tagged it when answering the help call. This approximation works well in practice for robots with differential drive systems, and in open environments. Other situations will require more sophisticated path planning methods. The

worst case target movement that is assumed for this computation of the time to capture makes sure that it is possible to recapture even intelligent targets which retrieve information on which robot may attempt to recapture them and move into the worst possible direction.

The introduced formalism allows to describe the behavior of B-CMOMMT in more depth. We assume an idealized scenario with a perfect target loss prediction mechanism that identifies the true time of loss for each target that is tagged by a particular robot. For all practical purposes a lower bound suffices, but for the theoretical analysis we are assuming perfect prediction.

Definition 6 (Target loss prediction) $t_l : O \times V \rightarrow \mathbb{R}$ *is the target loss prediction function that returns the predicted time of loss of a target* o_j *being tagged by robot* v_i*. In case robot* v_i *is not tagging target* o_j*, let* $t_l(o_j, v_i) = \infty$

The concept of perfect prediction is further fetched than it seems at first glance. Theoretically, it requires perfect knowledge about all trajectories of all targets and the computation of the trajectory of all robots. Hence the concept is merely useful for theoretical purposes. We will, however, introduce useful measures to transport some of the nice properties that perfect prediction grants us to practical scenarios with limited and erroneous prediction mechanisms (see section 6).

Definition 7 *Some events that occur during the execution of B-CMOMMT are defined as follows (let* $\delta > 0$*):*

1. Tag switch*: the event when* $[n_k, n_j] \in T(v_k, t)$ *and* $[n_l, n_j] \in T(v_l, t + \delta)$ *at* $t + \delta$ *with* $k \neq l$*.*
2. Target loss*: the event when* $[n_k, n_j] \in T(v_k, t)$ *and* $\nexists [n_l, n_j] \in T(t + \delta)$ *for any* l*.*
3. Target discovery*: the event when* $[n_l, n_j] \notin T(t)$ *for any* l*, and* $[n_l, n_j] \in T(t + \delta)$*.*

According to its definition, any change to the matrix $\Phi_{g(t)}$ is the result of one of the three above defined events: tag switch, target loss or target discovery. With regard to tag losses, no mechanism, except exploration, can recapture a target. The main benefit in B-CMOMMT is being able in certain circumstances to avoid target loss due to the help system. Under the assumption that robots are faster than targets, it is clear that no target followed by a robot exclusively can escape observation. Interesting questions arise when one robot follows multiple targets. Criteria for avoiding target loss in such a case are given by the following.

Theorem 1 (Re-capture a target: RC criteria) *Assume that a robot* v_i *observes multiple targets i.e.* $\sum_{j=1}^{n} \Phi_{g(t)}(i, j) > 1$*. If the following criteria are satisfied:*

1. *at least* $(\sum_{j=1}^{n} \Phi_{g(t)}(i, j)) - 1$ *robots are available for answering potential help calls successfully;*
2. *an assignment of these robots to the help calls exists s.t. all help calls are answered successfully;*
3. *no other help calls or targets interfere with the answering of help calls from* v_i *or another robot s.t criteria 1 and 2 remain satisfied,*

then all columns j *s.t.* $\Phi_{g(t)}(i, j) = 1$ *remain at column sum* 1*, i.e. the targets remain under observation by some robot.*

Proof: The first condition merely states that for every excess target o_j observed by v_i there is another robot v_k available, s.t. the time to capture T_r for robot v_k to capture o_j is less than the time of loss t_l predicted by v_i. The second condition ensures that we can answer all help calls with these available robots. Now once an assignment of help calls to robots that can re-capture the target of the help call is given, we only need to ensure that each of these re-capturing attempts are executed successfully. Since only targets and help calls can defer a robot from re-capturing the target of its assigned help call, we have to ensure that no other targets or help calls interfere. The third condition states exactly this. □

Let us now present the stable system theorem and its proof (in the following, $I_{n \times n}$ indicates the identity matrix of order n).

Theorem 2 (Stable system theorem) *Assume that $n = m$ and $\Phi_{g(t)} = I_{n\times n}$ at time t_0, i.e. every robot has tagged exactly one target. Then for all $t \geq t_0$ the sum of each column will remain 1, i.e. no target will get lost.*

Proof: Given $\Phi_{g(t)} = I_{n\times n}$, under the assumption that targets are slower than robots no target can escape its current pursuer. This means that we cannot have a target loss occurring as long as we have a one on one assignment of tags. The first event that can occur is a tag switch, i.e. one robot may drop its tag on a target, but the target is not lost yet because it is tagged by another robot. Now, let $\Phi_{g(t)}(i, j) = 1$ represent the tag from the i-th robot on the j-th target. Such a tag will only disappear if the tag is taken over by another robot, i.e. $\Phi_g(i', j) = 1$ for some other $i' \neq i$ and hence the column sum remains. Once we have one robot with more than one tag, the assumption that $s_v > s_o$ does not suffice anymore for ensuring that no target loss occurs. The main task of this proof is to show that despite a tag drop by one or more robots still no target loss will occur in the team.

Without loss of generality, let us consider the case when one or more tag changes occur from time t to $t + \delta t$. The tag changes result in k robots losing all their tags, and $m - k$ robots having m tags. If $k = 0$, then $\Phi_g = I_{n\times n}$ with a change of basis. Figure 8 illustrates this case. For $k > 0$ we have $l \geq 1$ robots that

Figure 8: A double tag switch occuring. In this case $k = 0$ because no robot remains without tags.

can issue k help calls. Each help call can be associated to a tag switch that created a zero row, namely for the robot that had a tag previously and is in explore mode now. All these k robots listen to help calls. On first sight it seems that with δt tending to zero we always have a robot v_l for each help call at a distance of less than $3d_{o_1}$ to the target for which the help call is issued. The target just lost by a now exploring robot is d_{o_1} away. The receiving robot has to be at distance less than d_{o_1} to the target it just received and d_{o_1} from the target it previously followed. Hence v_l is at most $3d_{o_1}$ away from the target associated with the help call. Figure 9 illustrates this case. There are, however, two problems with this simple scenario. It may be that the help call initiated moves further by multiple tag switches. Also the robot that just lost the target may not necessarily be available, i.e. it could have received another tag from another target that was lost by yet another robot. Even though such a situation is rather unlikely in practice, it is nonetheless possible as seen in figure 10. We now have to establish a bound on the worst case distance of the closest exploring robot to answer the help call. A concatenated tag switch occurs in an area with radius at most $2d_{o_1}r$ where r is the number of robots participating in the tag switch. Since the upper bound for admissible following distances d_{o_1} is adjusted to the number of robots and the sensing range (see equation 12), we know that the maximum distance d_h between exploring robot and help call associated to its tag loss is:

$$d_h \leq 2rd_{o_1} \leq 2r\left(\frac{r_s}{2(r+1)} - \xi\right) \leq \frac{r}{r+1}r_s - \xi \tag{14}$$

The factor ξ in this case reduces the total distance to accommodate for differential constraints on turning and accelerating. If none are present then $\xi = 0$. Having a bound on the distance suffices already to ensure capturing of the target. Our bound is tight enough s.t. the answering robot still has the target within sensing range and can turn into its direction. A relaxation on this part is, however, feasible since the predicted time of loss t_l for the help call is a lower bound. The center of gravity of all targets tagged by a robot v_i can at most move with the maximum target speed o_s. Hence a target can move away from this center of gravity by at most twice the target speed. Since d_{o_1} can be at most $1/4r_s$ at a tag switch (equation 12 with $r \geq 1$), we know that the target has to travel at least $3/4r_s$ from the center of gravity which gives us a time bound on the target loss.

Due to the perfect prediction, the time of loss for a help call remains constant since any change in the system is included in the perfect prediction. We now see that under the above circumstances, theorem 1 can be applied. The first of the three hypotheses is evidently satisfied. To show the second criterion holds we need to show that the available robots allow an assignment ensuring that all help calls are successfully

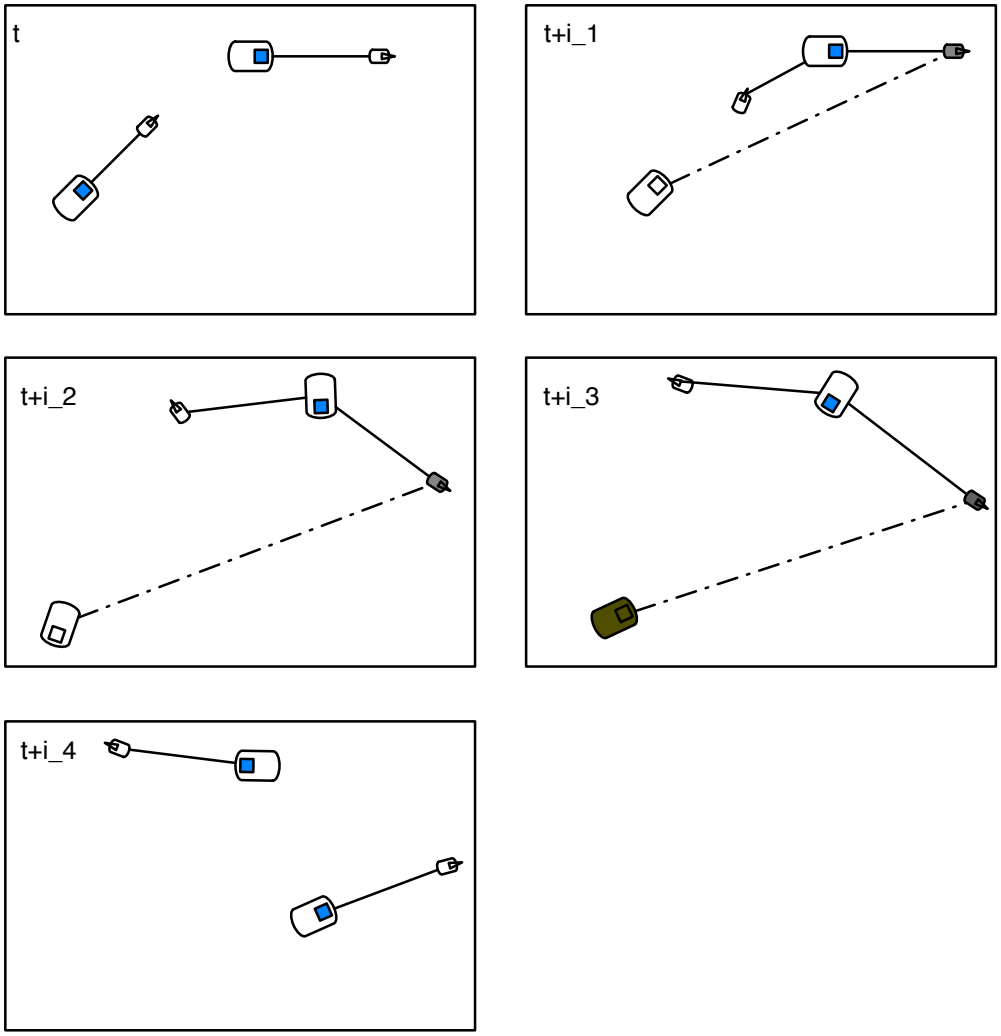

Figure 9: A usual tag change occurring. One target moves closer than d_{o_1} to another robot which gets two tags. In the second picture the robot that lost its tag becomes available for exploration and help purposes. It immediately answers the help call from the other robot and continues exploration until it is required to recapture the target. A solid line indicates tag ownership for a target, a dotted tag ownership for a help call. A grey robot is recapturing a target while all others are either following a target or exploring, depending on whether they own a tag. A grey target has a help call associated.

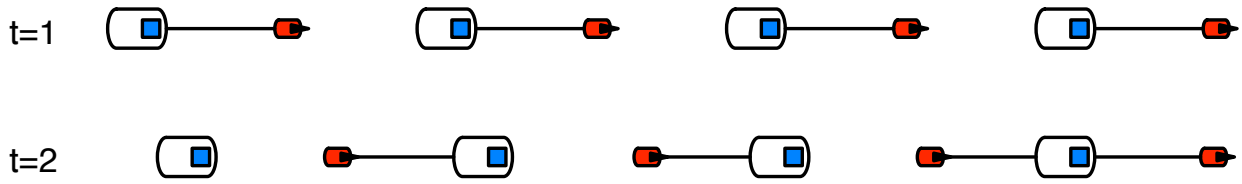

Figure 10: A concatenated tag change leaving the exploring robot far away from the robot with more than one tag. Solid lines indicate tag ownership.

answered. This is the linear bottleneck assignment problem for the help call assignment defined in section 4. In this case, the assumption that the input is a square matrix is satisfied naturally, since we have the same number of help calls and available robots. The input to the linear bottleneck assignment problem is

the matrix H which contains the time to capture for each available robot on each help call. As described in section 4, the time to capture is set to a high value or infinity when it is bigger than the time to loss.

$$H = \begin{pmatrix} T_r(v_{c'_1}, o_{c_1}) & \dots & T_r(v_{c'_1}, o_{c_k}) \\ \vdots & \ddots & \vdots \\ T_r(v_{c'_k}, o_{c_1}) & \dots & T_r(v_{c'_k}, o_{c_k}) \end{pmatrix} \tag{15}$$

A solution, if it exists, can be found in polynomial time (see [23]). We know that at least one solution exists s.t. all help calls are answered successfully, namely the one assigning each robot to the help call with bound on the distance. Once the assignment is completed only obtaining a tag may prevent a helping robot to recapture the target. A tag may be obtained by a tag switch or target discovery. Target discoveries cannot occur since $m = n$ and all targets are under observation. A tag switch cannot occur either since a robot recapturing a target does not tag targets that have a tag or an answered help call, unless of course the target is the one the robot is recapturing. Hence target loss does not occur.□

Corollary 3 *Under any type of target movement with a target to robot ratio* $\frac{n}{m} = R \geq 1$, $\lim_{t_e \to \infty} A = \frac{1}{R}$, *i.e every robot will always follow at least one target.*

Proof: This corollary follows directly from the proof of the stability theorem. All that may occur additionally is the discovery of targets which prevents the execution of the helping, but does not worsen the number of targets observed since it is irrelevant whether a new target is observed or the one that was previously to be captured with the helping.□

The introduction of centralized control for the assignment of help calls to robots may not be desired in some scenarios or simply not feasible. The necessity for such an assignment is, however, only given for the proof. In practice few help calls are issued at the same time and an assignment with a greedy tagging based on distances suffices. Furthermore, even with limited communication ranges the mechanism can work well since mostly robots that are close by qualify for the recapturing.

6 Experimental results

In order to validate the proposed BCMOMMT algorithm, extensive simulations have been performed with Player/Stage [6][20]. While it is inarguably true that experiments with real robots exhibit different challenges, the use of a widely used simulator like Player/Stage has the advantage that massive tests can be conducted with moderate resources. The experiments presented in the following should be interpreted as preliminary findings, and the collection of data with real robots is part of the outlook that will be presented in section 7. Overall 3,200 simulations with a duration between 2 and 4 minutes were run. The environment S was designed to be a circle with radius $30m$. As clarified later, robots and targets were initially deployed in a circular area of varying radius. Robots' maximum speed was set to $r_s = 0.4m/s$ and targets' maximum speed was $o_s = 0.3m/s$. Robot sensing range r_s was set to $5.5m$ or $4.0m$. Due to space limitations, not all results can be presented. The interested reader is referred to [12] for an exhaustive in depth discussion. Results concerning an early version of BCMOMMT without target loss predictions and modified help calls are instead presented in [13].

For the evaluation of the results it is of great value to observe the trend of the coverage across time. These observations provide a great deal of information about the temporal development of the algorithm and enable to isolate the factor of the initial deployment. These observations were lacking in [13] and [18]. The sets of experiments aim to give answers to the following questions:

1. How does the improved B-CMOMMT algorithm compare to the original B-CMOMMT, i.e. what is the impact of target loss prediction and improved help calls?
2. How does the theoretical result of full coverage with infinite runtime relate to practical simulations?
3. How significant is the effect of narrow initial placement?
4. How stable is the performance with many targets and few robots?

5. How do errors in target identification affect the performance?

Many more questions could be asked. For example, how to set the profile of the function rescaling the directional vectors, or how errors in localization, communication and prediction affect performance. We believe that all these questions are interesting in their own respect and deserve further investigations. It is however the case that these questions are better addressed in scenarios that move beyond the current simplified setting. Most importantly, the questions presented above already give us sufficient confidence about the behavior of B-CMOMMT to be able to continue into more challenging problem domains, adopting the basic ideas of the algorithm.

1. Performance of the improved B-CMOMMT algorithm. This first question can be answered by running the basic and the improved B-CMOMMT algorithm under the same conditions, including robots and targets starting points. To allow a direct comparison three experimental sets have been run with identical settings. Results are summarized in figure 11 for different values of the target to robot ration n/m. Each point shows the average coverage obtained in 100 simulated runs. The improvements show that the modifications of the help calls and target loss prediction are indeed valuable.

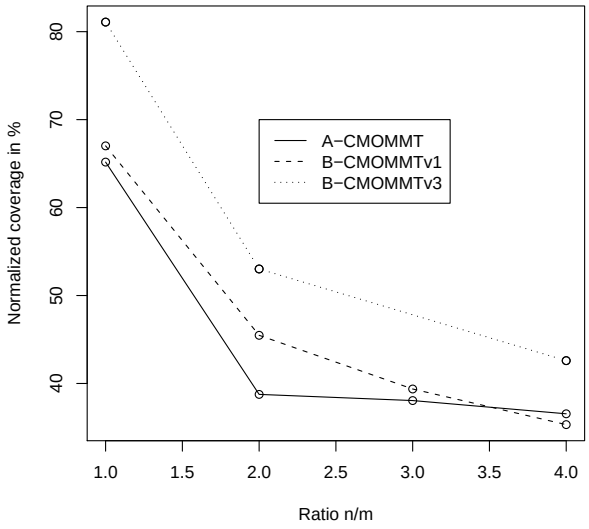

Figure 11: Comparison of the basic and improved B-CMOMMT algorithm.

2. Theoretical results in practice. According to the main theorem of section 5, under suitable conditions, once a situation with a one on one assignment and perfect prediction are in place, full coverage persists. Practically, perfect prediction is impossible and errors in prediction will always have some effect. Furthermore, simulations start with a random placement and a one on one assignment is not likely to occur early on. Finally, we are using the distributed assignment of help calls. The experimental setup therefore relies on different and *weaker* conditions than the proven theorem. As first experiment, a narrow initial placement of few robots and targets was set, in order to make a one on one initial placement or at least early adjustment more likely. In figure 12 the development of the coverage across time is shown. The narrow initial placement grants full coverage at first. Some targets, those at the boundary, however, manage to escape. On average one target escapes every four runs dropping the average coverage to around 94%. After a short period of adjustment, robots manage recapture some of these targets. Note that a target which is in the predictive tracking range of a robot is not counted towards the coverage, since it is not identified by the sensor, although its position continues to be estimated. This increase of coverage is not only observed in the case of narrow initial placement. In figures 13, 14 and 15 we can also observe an increase in coverage across time. These figures present experimental sets with ten robots and ten targets with increasing radius of the initial placement. Targets that are gradually discovered are hence kept under surveillance. Figure 12

and 13 suggest that finally a constant coverage of around 95% can be achieved. The coverage is close to full, but due to the limitations with regard to the prediction a value of 100%, is not reached, which is somehow expected.

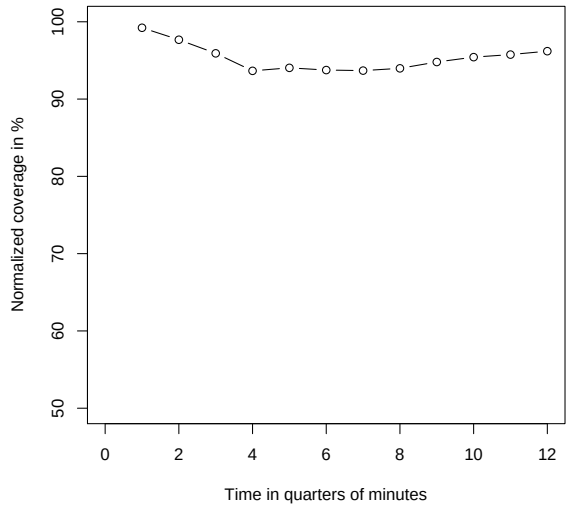

Figure 12: Development of coverage across time with all targets and robots initially placed in a circle of radius $5m$ ($m = n = 10$).

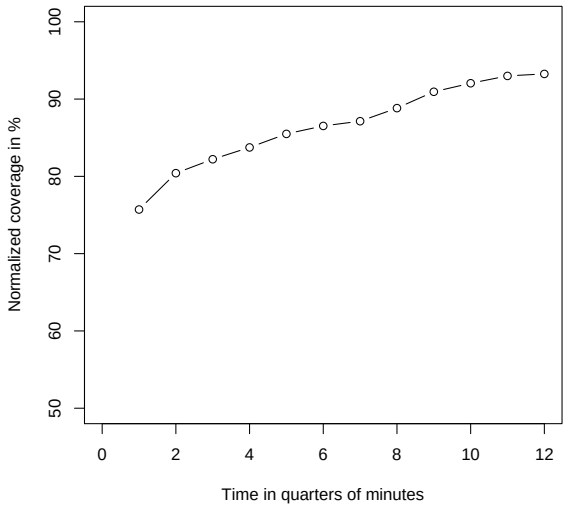

Figure 13: Development of coverage across time with all targets and robots initially placed in a circle of radius $15m$ ($m = n = 10$).

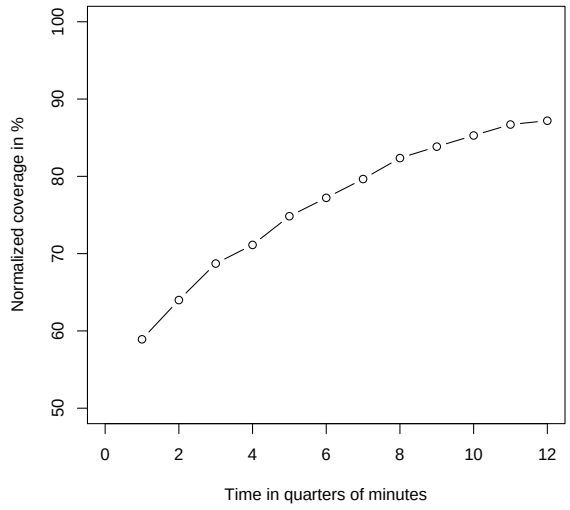

Figure 14: Development of coverage across time with all targets and robots initially placed in a circle of radius $20m$ ($m = n = 10$).

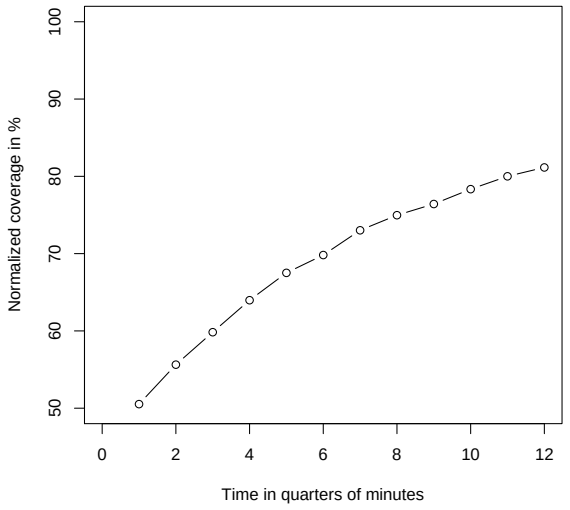

Figure 15: Development of coverage across time with all targets and robots initially placed in a circle of radius $25m$ ($m = n = 10$).

3. Significance of the initial placement. The initial placement has a considerable effect on the coverage. An improved performance for a narrower initial placement can be expected. It is nevertheless

interesting to see how the coverage develops differently across time in each of these cases. Hence, we ran experiments that correspond to figures 12, 13, 14 and 15 with an extended simulation time of up to 6 minutes. Furthermore, we added another case where the initial radius of the deployment is 10m. The results are plotted in figure 16. Under perfect circumstances, i.e. satisfying the prerequisites of the theorem, and with infinite simulation time we would expect to approach a coverage of 100% in all cases. The results already show how the differences between the different radii become smaller. They suggest that a coarser placement requires more time for the algorithm to adjust, but also that it manages to adjust with a certain time delay.

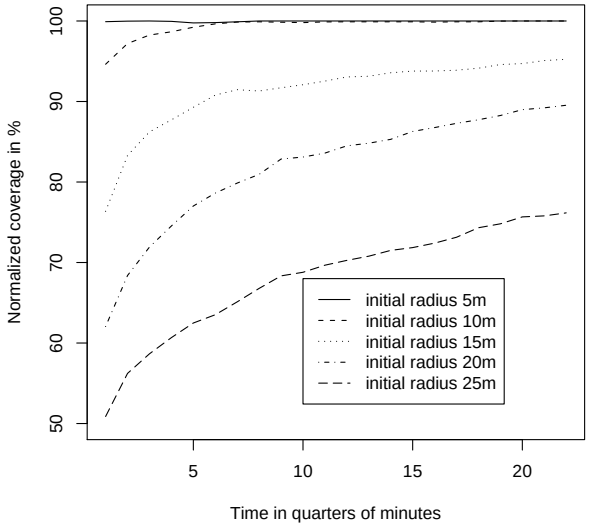

Figure 16: Comparison across time between B-CMOMMT for initial placements with radii 5m, 10m, 15m, 20m and 25m respectively with an extended simulation time to 6 minutes.

4. Performance with many targets. A major achievement for an algorithm solving the CMOMMT problem is the ability to observe many targets with few robots. When limited resources do not allow to aim for full coverage, it is desirable to ensure that B-CMOMMT manages the robots efficiently. For this purpose we shall introduce an additional performance criterion that shows how well the algorithm utilizes each robot in the presence of many targets. Let us call it *utilization* and let it be the normalized coverage multiplied with the target to robot ratio. We have seen in figure 11 that our improvements outperform our previous approach for target to robot ratios of one, two and four. Such plot, however, only shows the average value for two minute runs, as done in a previous publications on the topic. Furthermore, we cannot only look the target to robot ratio. A target to robot ratio of one may mean ten robots and ten targets or five robots and five targets. These two may lead to completely different results as we can see from figure 17. We see the same pattern when we increase the radius of the initial placement. This is quite obvious since the team with ten robots can cover the initial area better as the one with five. If we were to decrease the radius for the team of five s.t. the area is half that of the one for the team of ten we may obtain identical results and thereby clarify the relationship between the robot to target ratio, the number of robot and the radius of initial placement. This new hypothesis could only be confirmed by further simulations as a follow up to what is presented here. To gain further insights in the performance with many targets we performed additional experiments with different target to robot ratios. Results are summarized in figure 18. The difference in coverage for the entire time is not statistically significant. Increasing the team from 10 to 15 robots hence leads to only a minimal increase in performance. Figure 19 with the plot of the utilization confirms this. Still, each additional robot increases the coverage, even if only slightly, and there seems to be no saturation. This is expected with evasive target movement since any robot that follows two targets will eventually lose one if no other robot is available for help. The task of the algorithm is to maximize the time such a loss can be avoided. The only way to reach a saturation is simply by covering the entire area or developing a

new algorithm and design it for evasive target movement. The design would be such that the robots drive the targets, just like a shepard dog guides sheep, in such a way that they are constantly surveilled. Whether such an algorithm can be conceived is a whole other question. This sort of task lies very close to the realm of non-cooperative game theory and is out of the scope of this paper.

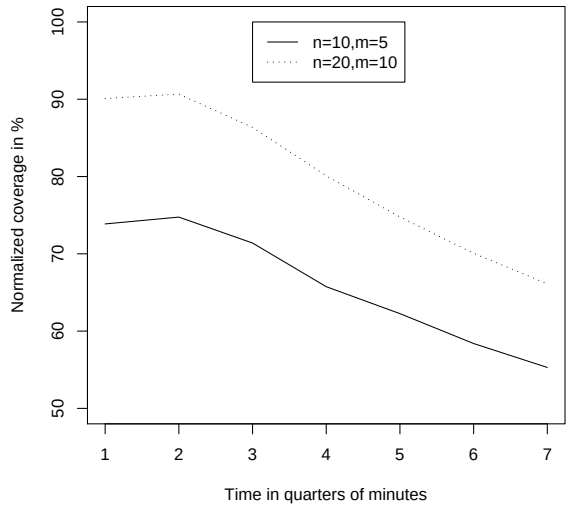

Figure 17: Two sets with identical target to robot ratios, but different number of robots.

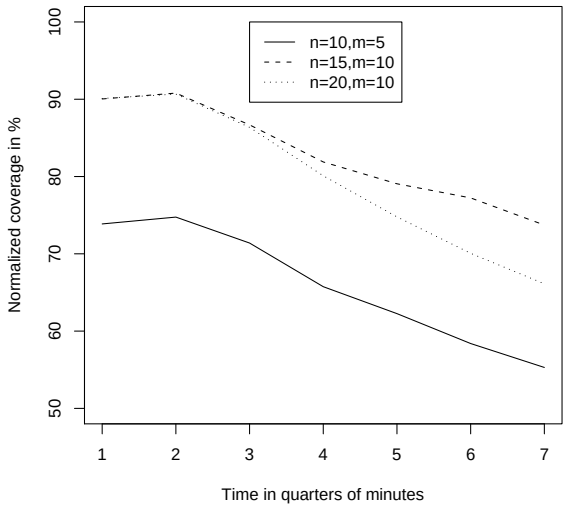

Figure 18: Experimental sets with more targets than robots.

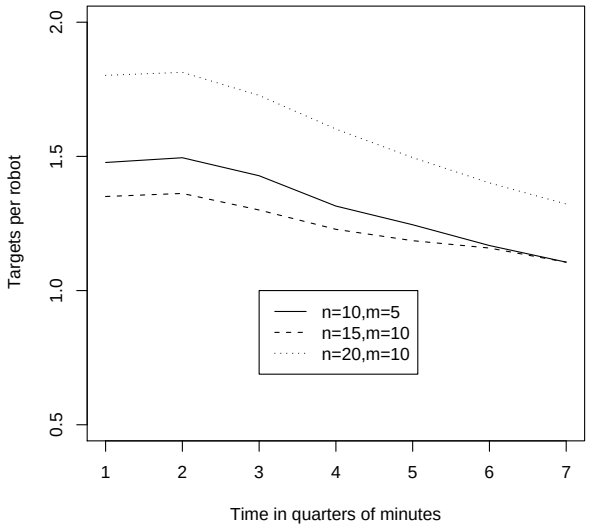

Figure 19: Experimental sets with more targets than robots are compared with regard to the utilization of each robot. Utilization is usually higher for a higher target to robot ratio.

5. Errors in target identification. The whole set of questions concerned with the influence of errors has received relatively little attention up to now. In its distributive and suboptimal nature the algorithm

has a built in compensation for errors. If a robot fails to operate properly, others will not be influenced drastically. Errors in localization and sensing will certainly have some effect, but only to the degree that they will prohibit a proper following or lead to considerably misestimated times for the prediction of target loss. Knowing the error range for the estimation, however, allows us to set our thresholds so that helping is carried out earlier. We shall at least look at one particular error and its effect on the algorithm, namely the error in target identification. Three sets of experiments have been dedicated to the investigation of the effect of erroneous target identification. Advanced target identification approaches such as in [1] and [17] which provide methods to solve the data association problem for sensor observations can reduce such errors. Despite this, depending on the implementation such identification may have exceeding error rates that could inhibit the proper functioning of B-CMOMMT. Figure 20 summarizes the results coming from simulations where error rates of 10%, 30% and 50% have been introduced. These are compared with error free simulations. The error was introduced by swapping the real target id with probability 0.1, 0.3 and 0.5 with another random target id. No measures in the clients were taken to deal with cleaning up the communication channels for the just previously observed target and help calls are left intact. It turned out that the effect of the swapping, be it with 0.1, 0.3 or 0.5 is equally drastic. Whether the robot swaps an expected number of 10 or 50 targets while actually following one and the same target does not change the impact of the error. It is easy to imagine that already one wrong broadcast for a target, e.g. claiming a tag wrongfully, can lead to a loss of the target when the robot who actually should follow that target drops it. Another wrong broadcast on this target will not have any effect since the target is already lost out of observation. Although the nature of the introduced error does not resemble any realistic error, the fact that errors can have a drastic effect remains and should not be taken lightly. It is, however, advisable to focus on this important area when moving to a more complicated problem domain, including more complex environments and realistic exploration. The effect of errors in localization, sensing and prediction will be more drastic in complex indoor environments.

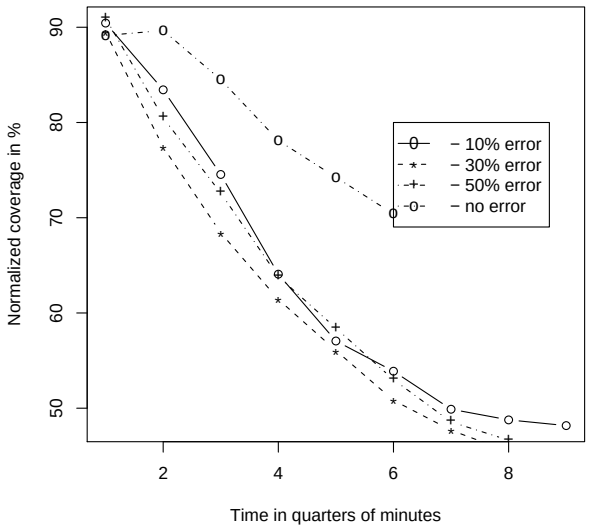

Figure 20: Performance under varying error in the identification of targets for errors of 10%, 30% and 50%, m=10, n=15.

7 Conclusions

In this paper we have presented a solution to the CMOMMT problem based on cooperation through communication. Our formerly developed B-CMOMMT algorithm schedules three mutually exclusive behaviors on the pursuer robots, namely *follow*, *explore* and *help*. B-CMOMMT has been here extended introducing more refined techniques for target loss prediction, and a better way for answering help calls. The proposed

modifications yield significant improvements in terms of observed performance. Extensive simulations help to better understand the importance of the different elements building the proposed algorithm. For example, the issue with perfect prediction of time of loss is not as drastic as it may seem. Perfect prediction may be difficult to achieve, but the errors in prediction can be counteracted by answering help calls earlier, i.e. setting h_m to a higher value. There are, however, some errors for which the effect is not yet well understood. This refers in particular to the sensitivity to communication and target identification errors. B-CMOMMT attempts to make best usage of the given information and no error handling is incorporated into this as of now. Hence errors may have a fatal effect as seen section 6 in the runs with errors in target classification. It remains to investigate properly what type of errors have an effect on the working of B-CMOMMT. This, however, should be attempted when path planning, map building and complex environments are dealt with, since this is when further and possible more fatal errors come about. Moreover, in this paper we have presented some theorems proving that the B-CMOMMT algorithms bears a stable performance. This kind of analysis and proof is novel in the context of the CMOMMT problem, and more generally for algorithms scheduling different behaviors within multirobot systems. In the future, we intend to investigate this problem in more complicated environments, and introducing further restrictions on the sensor payload. Investigation with physical robots are also planned.

Acknowledgments

This paper extends preliminary results presented at the 2006 IEEE International Conference on Robotics and Automation [13].

References

[1] Y. Bar-Shalom and T. E. Fortmann. *Tracking and data association.* Academic Press Professional, Inc., San Diego, CA, USA, 1987.

[2] W. P. Chin and S. Ntafos. Watchman routes in simple polygons. *Discrete and Computational Geometry*, pages 9–31, 1991.

[3] J. Cortés, S. Martínez, and F. Bullo. Robust rendezvous for mobile autonomous agents via proximity graphs in arbitrary dimensions. *IEEE Transactions on Automatic Control 2004*, 51(8):1289–1298, 2006.

[4] J. Cortés, S. Martínez, T. Karatas, and F. Bullo. Coverage control for mobile sensing netorks. *IEEE Transactions on Robotics and Automation*, 20(2):243–255, 2004.

[5] B. Gerkey, S. Thrun, and G. Gordon. Visibility-based pursuit-evasion with limited field of view. *Internationla Journal of Robotics Research*, 25(4):299–316, 2006.

[6] B.P. Gerkey, R.T. Vaughan, and A. Howard. The player/stage project: Tools for multi-robot and distributed sensor systems. In *Proceedings of ICAR*, pages 317–323, 2003.

[7] H. H. González-Baños, L. Guibas, J.-C. Latombe, S. M. LaValle, D. Lin, R. Motwani, and C. Tomasi. Motion planning with visibility constraints: Building autonomous observers. In Y. Shirai and S. Hirose, editors, *Proceedings Eighth International Symposium on Robotics Research*, pages 95–101. Springer-Verlag, Berlin, 1998.

[8] 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 and Applications*, 9(4/5):471–494, 1999.

[9] A. Jadbabaie, J. Lin, and A.S. Morse. Coordination of groups of mobile autonomous agents using nearest neighbor rules. *IEEE Transactions on Automatic Control*, 48(6):988–1001, 2003.

[10] B. Jung and G. Sukhatme. Tracking targets using multiple mobile robots: the effect of environment occlusion. *Autonomous robots*, 13(2):191–205, 2002.

[11] B. Jung and G. Sukhatme. Cooperative multi-robot target tracking. In M. Gini and R. Voyles, editors, *Proceedings of the 8th International Symposium on Distributed Autonomous Systems*, pages 81–90. Springer-Verlag, 2006.

[12] A. Kolling. Multirobot cooperation for surveillance of multiple moving targets – an improved behavioral approach and its formalization. Master's thesis, International University Bremen, Germany, 2006. Also available as IUB Technical Report N. 03.

[13] A. Kolling and S. Carpin. Multirobot cooperation for surveillance of multiple moving targets - a new behavioral approach. In *Proceeding of the IEEE International Conference on Robotics and automation*, pages 1311–1316, 2006.

[14] S. M. LaValle, D. Lin, L. J. Guibas, J.-C. Latombe, and R. Motwani. Finding an unpredictable target in a workspace with obstacles. In *Proceedings IEEE International Conference on Robotics and Automation*, pages 737–742, 1997.

[15] S. Martínez and F. Bullo. Optimal sensor placement and motion coordination for target tracking. *Automatica*, 42(4):661–668, 2006.

[16] D. L. Mills. Improved algorithms for synchronizing computer network clocks. *IEEE Transactions on Networking*, 3(3):245–254, June 1995.

[17] S. Oh, S. Russell, and S. Sastry. Markov chain monte carlo data association for general multiple-target tracking problems. In *Proceedings of the 43rd IEEE Conference on Decision and Control*, 2004.

[18] L. E. Parker. Distributed algorithms for multi-robot observation of multiple moving targets. *Autonomous robots*, 12(3):231–255, 2002.

[19] L.E. Parker. Alliance: An architecture for fault-tolerant multi-robot cooperation. *IEEE Transactions on Robotics and Automation*, 14(2):220–240, 1998.

[20] Player/stage project. http://playerstage.sourceforge.net, 2005.

[21] S. Sachs, S. Rajko, and S. M. LaValle. Visibility-based pursuit-evasion in an unknown planar environment. *International Journal of Robotics Research*, 23(1):3–26, 2004.

[22] M. Saptharishi, C. Spence Oliver, C.P. Dihel, K.S. Bhat, J.M. Dolan, A. Trebi-Ollennu, and P.K. Kohsla. Distributed surveillance and reconnaisance using multiple autonomous atvs: Cyberscour. *IEEE Transactions on Robotics and Automation*, 18(5):826–836, 2002.

[23] L. Schenato, S. Oh, S. Sastry, and P. Bose. Swarm coordination for pursuit evasion games using sensor networks. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2493–2498, 2005.

[24] T.C. Shermer. Recent results in art galleries. *Proceedings of the IEEE*, 80(9):1384–1399, 1992.

[25] I. Suzuki and M. Yamashita. Searching for a mobile intruder in a polygonal region. *SIAM Journal of Computing*, 21(5):863–888, 1992.

[26] B.B. Werger and M.J. Mataric. Broadcast of local eligibility for multi-target observation. In Parker et al., editor, *Distributed autonomous robotics systems 4*, pages 347–358. Springer, 2000.