Master Thesis

# Design of a Protocol for event-based Network Reconfiguration of Active Vision Systems

Hannover, April 5, 2011

Author: Alvaro del Amo
First examiner: Prof. Dr. Jörg Hähner
Advisor: Dipl.-Ing. Wittke, Michael, M. Sc.

# Preface

This report is the result of my master thesis project carried out in the Institut f*ü*r Systems Engineering (System und Rechnerarchitektur) in Leibniz Universt*ä*t Hannover.
Camera sensor networks are becoming increasingly important to next generation applications in surveillance, in environment and disaster monitoring, city and traffic management and in the military.
Inside the robotics community the cooperative multi-robot observation of multiple moving targets (CMOMMT) problem and similar tasks enjoyed significant popularity, also because they naturally call for coordination and cooperation.

This Master Thesis implements a self-optimising system architecture for networked smart cameras and shows that cooperation between mobile smart cameras, and organic node computing really make a big difference.

I would like to thank the following persons: Prof. Dr. J*ö*rg H*ä*hner for being my supervisor and Examiner, Dipl.-Ing. Michael Wittke, his supervision, his patience and insight were key to the completion of this project. All the S.R.A. partners, especially to Sascha Radike for discussions and comments regarding the thesis and for their company. Also thanks to all old and new friends, my parents and Cristina Navarro for all the support they gave me.

# Abstract

Nowadays surveillance of large areas, such as banks, airports or cities is mostly based on vision systems. Smart Cameras (SCs) play an important role for security systems as they combine video sensing, video processing and communication within a single device. One weak point is that SCs are usually stationary and so occlusions may create blind spots in the system.
These blind spots may be overcome by using mobile SCs, so called Active Camera Networks (ACNs), as introduced in this thesis.

Nevertheless, mobility of SCs come along with challenges in terms of coordination and configuration. In addition to this, the cost of ACs is higher in comparison to static camera networks but the number of guards needed to survey a large area like a city or to control a lot of monitors can be reduced considerably with AC networks.

Our goal is to implement a self-reconfiguration system architecture for networked smart cameras that could be mounted either on mobile robots on the ground or Micro Air Vehicles (MAVs). Thus, the ACs will decide by themselves where to update their position in order to achieve the optimal system's performance. To reach that goal, ACs will increase or decrease spatial redundancy regions with their neighbours to overcome overloaded regions. The protocol presented in this thesis adapts the position of the ACN to the different trajectories that traverse a surveillance area over time.

The simulations have shown that the presented protocol increase the overall performance due to the node reconfiguration and cooperation between neighbouring ACs.

# Abbreviations

**AcViS Active Video Systems**

**AC Active Cameras**

**AoM Area of Movement**

**CoM Center of Movement**

**CMOMMT Cooperative multi-robot observation of multiple moving targets**

**DSC Distributed Smart Camera**

**FoV Field of View**

**MSC Mobile Smart Camera**

**ToIs Targets of Interest**

**UF Utility Function**

# Contents

# 1 Introduction

## Contents

Camera sensor networks are becoming increasingly important to next generation applications in surveillance, in environment and disaster monitoring, city and traffic management and in the military. In contrast to current video surveillance systems, camera sensor networks are characterized by smart cameras, larger network sizes, and ad hoc deployment. These systems lay in the intersection of machine vision and sensor networks, raising issues in the two fields that must be addressed in unison. The effective visual coverage of extensive areas -urban environments, disaster zones, and battlefields- requires multiple cameras to collaborate towards common sensing goals. As the size of networks grows, it becomes infeasible for human operators to monitor the multiple video streams and identify all possible events. Therefore, it is desirable to develop camera sensor networks that are capable of performing visual surveillance tasks autonomously or at least with minimal human intervention. This can be achieved with the help of sensor position networks.

## 1.1 Background

Nowadays surveillance of large areas, such as banks, airports or cities is mostly based on vision systems. Smart Cameras (SCs) play an important role for security systems as they combine video sensing, video processing and communication within a single device. One weak point is that SCs are usually stationary and so occlusions may create blind spots in the system.
These blind spots may be overcome by using mobile SCs, so called Active Camera Networks (ACNs), as introduced in this thesis.

Nevertheless, mobility of ACs come along with challenges in terms of coordination and configuration. In addition to this, the cost of ACs is higher in comparison to static camera networks but the number of guards needed to survey a large area like a city or to control a lot of monitors can be reduced considerably.
Another interesting point is that with ACNs, events can be detected in advance, for example including some kind of alerts to suspicious situations, such as people running, hiding, etc.
Additionally, mobility can improve the imaging quality of the system through the following aspects:[AKS07]

1. *Phenomenon Adaptivity:* Mobile ACs can move to sample the phenomenon precisely where the highest resolution coverage is required.

2. *Medium Adaptivity:* Mobile ACs can reposition themselves to overcome medium obstacles and anisotropies.

3. *Increased Sensor Range:* Mobile cameras can cover a larger volume depending on the acceptable motion delay.

4. *Robust Reconfiguration:* If one or more cameras fail, i.e. due to mechanical failures, neighbouring nodes can update their pose to achieve an optimal global field of view.

The objective of this thesis is to design a protocol making way for self-learning and self-configuration in Active Camera Networks as described in the following section.

## 1.2 Problem Description

The objective of this master thesis is to design a protocol for the event-based reconfiguration of Active Camera Networks. The performance of the proposed approach has to be proved by simulation due to the cost of the infrastructure needed. The first contribution of this master thesis is to study different protocols for distributed Smart Camera systems, and the design of a new one considering mobility of cameras.

The goal of this master thesis was to:

- Design of a Protocol for event-based Network Reconfiguration of Active Vision Systems

- Analyze the performance of the protocol theoretically and by simulation.

- To create reusable code structures, hardware, and research methodologies providing the foundations for further studies into event-based reconfiguration protocols.

## 1.3 Related Work

There has been a significant amount of research devoted to the use of sensors and intelligent devices for surveillance. But a complete coverage is not beyond the scope of this thesis. Therefore, we only discuss related work using cooperative approaches for visual surveillance.

### 1.3.1 Cooperative Observation

Cooperation and target detection under a surveillance area, is closely related to the *art gallery* problems. O'Rourke et al. [O'R87] investigates how to minimize the number guards required to completely surveill an interior polygonal area. Focusing on different variations of the problem; fixed guards and mobile guards that patrol following straight trajectories.

Hoffmann et al. [HH07] propose an algorithm were SCs are able to decide autonomously how to arrange their fields of view (FOVs) for maximizing spatial surveillance coverage. Each SC uses a PTZ camera and they locally adjust their FOV and exchange state information with other cameras in the network. The surveillance area can be divided in Smart Camera Sub-Systems (SCSSs). The goal of ROCAS protocol is to get an optimal partitioning of a surveillance area, so that the surveillance coverage becomes maximal. The heuristic bases upon the assumption, that minimising overlap locally in the SCSSs leads to a global minimum in the whole area, too.

Parker et al. [Par02b] investigates the use of a cooperative team of autonomous sensor-based robots for the observation of multiple moving targets. Focusing on developing distributed control strategies that allow a team of robots to minimize the total time in which the targets escape the surveillance area by some robot team member in the area of interest, given the locations of nearby robots and targets. The analysis of this approach is compared with fixed SCs, robots with random movements and with robots with local knowledge.
The goal of the robots is to maximize the average number of targets in the surveillance area that are being detected by at least one robot during a period of time T. The robots can communicate to know the state of their neighbours, and also receive the speed and pose of the targets to compute the decision of their new pose. The algorithm approaches the problem, with robots that are repelled by neighbouring robots and attracted by neighbouring targets, with more or less strength depending on the speed and direction of the other elements. With the final goal of maximal coverage and target detection.

Kolling et al. [AK06] presents a behaviour-based solution to the problem of CMOMMT sharing the workload in the surveillance area. Robots sense targets using sensors and in addition exchange information about them with other robots. The goal is to get maximal target coverage so each target must be observed by at least one robot. Robots may explicitly ask for help if they realize that some target will soon escape their observation, and they also can provide support to robots asking for assistance.
More precisely, their approach divide the surveillance area in different regions and assigns different robots to each to track targets in it. The reaction of the robots inside these regions is based on the vector update explained by Parker et al. In addition to this, the robots have

three different states; follow, help and explore mode.

### 1.3.2 Self-learning by Phenomenon Adaptativity

Kansal et al. [AKS07] presents an architecture that allows each node in the network to learn the medium and phenomenon characteristics. Shows a distributed optimization algorithm which computes a desirable network configuration and adapts it to environmental changes. The presented network protocol implements a distributed motion control algorithm, and faces practical situations such as communication needed for coordination and optimal motion without global coordination. The SCs using motility (pan, tilt and zoom) cover the area with lower resolution, and detects the targets with high resolution sensing.
When multiple sensors can observe a point p, an utility function that consider points under range and outranged, is used to determine which sensors must detect the events and how many if the regions receive more events that were expected.

Wu et al. [ASW99] focus on the surveillance of large areas with Micro Air Vehicles teams (MAVs) with sensors. The approach, dynamically distributes the MAVs in the surveillance area looking for maximal coverage based on features present on the ground, and to adjust the distribution if trajectories or changes in the MAVs teams occur. Each MAV follows basic rules that place them in the optimal pose in a distributed manner. Computing which is the best rule in their set for a given situation.

Jund et al. [JS02] introduce a Region-based Approach which controls robot deployment at two levels. A deployment controller distributes robots across regions using a topological map which maintains urgency estimates for each region, and a target-following controller attempts to maximize the number of tracked targets within a region.
Given an scenario, each robot computes two density estimates for each region. The *Robot Density* is the number of robots in a region normalized by the sensing area of a robot. And the *Target Density* that estimates the number of targets in each region normalized by the sensing area of a robot. Each AC attempts to track as many targets as possible within its current region, and periodically computes if it must move to other regions where more robots are needed.

### 1.3.3 Swarm Intelligence

Another approach to face the CMOMMT problem is swarm intelligence. Biologists rely on locational optimization tools to study how animals share territory and to characterize the behaviour of animal groups obeying the following interaction rule: each animal establishes

a region of dominance and moves toward its center.
The relation between Self-organization systems and Swarm Robotics is presented in Trianni et al. [VTD08]. Where is presented an overview of successful applications of evolutionary techniques to the evolution of self-organising behaviours for a group of simulated autonomous robots.

Hsiang et al. [KHLMHA04] approach uses the rules of ant behaviour to self-configure the regional distributions of sensors in proportion to the number of the moving targets that need to be observed in a non-stationary environment. The robots only use local sensing and short range communication. The information about targets and other robots in a region is computed due to the waiting time a robot spends until it encounters other robots and targets.
The decision to move to a region or to stay in current location, depends on the ratio of the waiting times of each region. In addition to this, each time a robot has an encounter with another robot, they must contest for the target, increasing their probability to stay in the current region or to leave. Communication between robots is needed, as the information about the waiting times in the other regions is only achieved by travelling to this regions or if some robot from this area travels to another area broadcasting its knowledge of the system.

# 2 General Concepts

## Contents

## 2.1 Smart Cameras

Rinner et at. [RW08], explains that Smart Cameras are smart visual sensors in a single embedded platform which can achieve three different important tasks:

1. Video sensing.
2. Image processing.
3. Communication.

Due to this reason multi-camera applications or scenarios can work in a decentralized manner. This way the image processing does not depends on a central workstation as each camera performs its own processing. By SC systems the communication load is reduced and the reliability and scalability of multi-camera applications is increased.

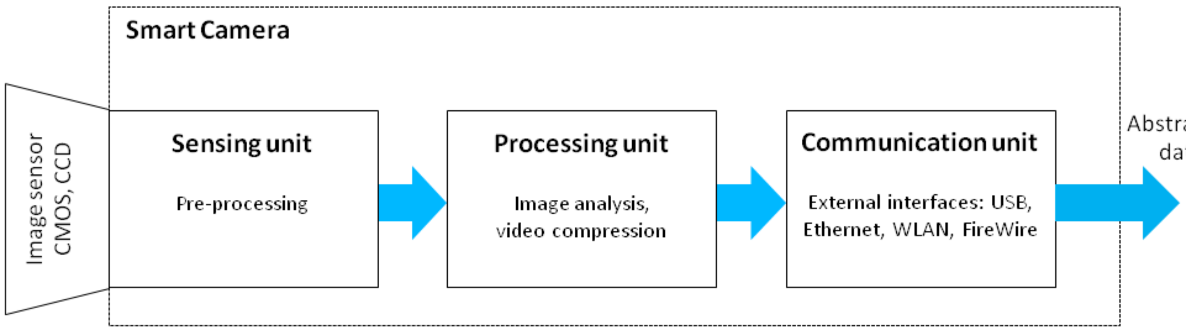

Figure 1: Generic Smart Camera

### 2.1.1 Architecture

A generic architecture of a smart camera comprised of a sensing, processing and communication unit is depicted in Figure 1. The image sensor, which is implemented either in

CMOS or CCD technology, represents the data source of the processing pipeline in a smart camera. The sensing unit reads the raw data from the image sensor and often performs some preprocessing such as white balance and color transformations. This unit also controls important parameters of the sensor, e.g., capture rate, gain, or exposure, via a dedicated interface. The main image processing tasks take place at the processing unit which receives the captured images from the sensing unit, performs real-time image analysis and transfers the abstracted data to the communication unit.
The communication unit controls the whole processing pipeline and provides various external interfaces such as USB, Ethernet or Firewire. These generic units are implemented on various architectures ranging from system-on-chip platforms over single processor platforms to heterogeneous multi-processor systems. Field programmable gate arrays (FPGAs), digital signal processors and/or microprocessors are popular computing platforms for smart camera implementations. The main design issues for building smart cameras are to provide sufficient processing power and fast memory for processing the images in real-time while keeping the power consumption low.
Smart cameras perform a variety of image processing algorithms such as motion detection, segmentation, tracking, object recognition and so on. They typically deliver color and geometric features, segmented objects or rather high-level decisions such as wrong way drivers or suspect objects. The abstracted results may either be transferred within the video stream, e.g., by color-coding, or as a separate data stream. Note that the onboard computing infrastructure of smart cameras is often exploited to perform high-level video compression and only transfer the compressed video stream.

### 2.1.2 Applications

Current and potential applications of Smart Camera Networks include: Intelligent Transportation Systems, Medicine [MLY04], Machine Vision, and Intelligent Video Surveillance Systems (IVSS).

IVSS is a very active research area. The fundamental goal in IVSS is to detect “abnormal” behaviours in the observed scene [CRF01],[VV05]. This requires complex image analysis starting from motion detection to segmentation, feature extraction, and classification. To extend the spatial sensor coverage IVSS are typical examples for multicamera systems. However, most traditional IVSS require no or only little cooperation among individual cameras.

### 2.1.3 Distributed Smart Cameras

The term distributed camera refers in computer vision to a system of physically distributed cameras that may or may not have overlapping fields of view. The images from these cameras are analyzed jointly. Distributed cameras allow us to see a subject of interest from several different angles. This, in turn, helps us solve some very hard problems that arise in single-camera systems. Occlusion is a major problem in single-camera systems. When we have multiple views of a subject, we are much more likely to be able to see the parts of an object occluded in one view by switching to another camera's view.

Another way to think about distributed cameras is pixels on target. Our ability to analyze a subject is limited by the amount of information, measured in pixels, that we have about that subject. Not only do distributed cameras give us several views, but one camera is more likely to be closer to the subject. A traditional camera setup would use a single camera to cover a large area. Subjects at the opposite end of the space would be covered by very few pixels. Distributed camera systems help us cover the space more evenly. For example in our thesis, when a target is detected, the closer idle camera will take action, making faster the reaction time.

Occlusion may be static or dynamic. A fixed object, such as a wall or a table, causes occlusion problems that are easier to predict. When one moving object occludes another, such as when two people pass each other, occlusion events are harder to predict. Due to mobility the cameras can avoid this occlusion and get the needed data changing its pose.

The number of cameras we need to cover a space depends on both the field of view of the camera and the required number of pixels on target. For a typical camera with a rectangular image sensor, the field of view is a pyramid extending from the lens. The angular field of view of the lens determines the size of this pyramid. A normal lens provides the same angular field of view as does the human eye, between 25°and 50°. A wider lens covers more area in the scene, spreading a given number of pixels over a larger area in the scene. A longer lens covers less area in the scene, putting more pixels on the target. The pixels-on-target criterion and the size of the smallest target of interest tell us how far this pyramid extends from the camera. A distributed computer is a network of processors in which the nodes (processors) do not have direct knowledge of the state of other nodes. A node can retrieve the state of another node only by receiving a message from that node. Messages in distributed systems have nontrivial costs, so distributed algorithms are designed to minimize the number of messages required to complete the algorithm.

Distributed computers introduce several complications. However, we believe that the problems they solve are much more important than the challenges of designing and building a distributed video system. As in many other applications, distributed systems scale much more effectively than do centralized architectures. Many realistic applications require enough cameras that server-based architectures are not realistic.

Processing all the data centrally poses several problems. Video cameras generate large quantities of data. If we transmit raw video to a server, the network must be able to handle the required bandwidth in steady state. Furthermore, the server itself must be able to handle such large data quantities and move it from the network interface through the memory, into the processor, and out to mass storage. Moving video over the network also consumes large amounts of energy. In many systems, communication is 100 to 1000 times more expensive in energy than computation.
Although data must be compared across several cameras to analyze video, not all pairs of cameras must communicate with each other. If we can manage the data transfer between processing nodes, we can make sure that data only go to the necessary nodes. A partitioned network can protect physically distributed cameras so that the available bandwidth is used efficiently.
Real-time considerations also argue in favor of distributed computing. The round-trip delay to a server and back adds to the latency of making a decision, such as whether a given activity is of interest. A distributed system can make sure only relevant nodes are involved in a given decision. To the extent that we avoid sharing common resources, we also increase the predictability of processing time.

## 2.2 System Model

In this section we present a set of assumptions taken to characterize the system in which Active Cameras (ACs) operate and which comprise the system model. Additionally, we present an overview of the anticipated system architecture for dynamic reconfiguration in AC networks.

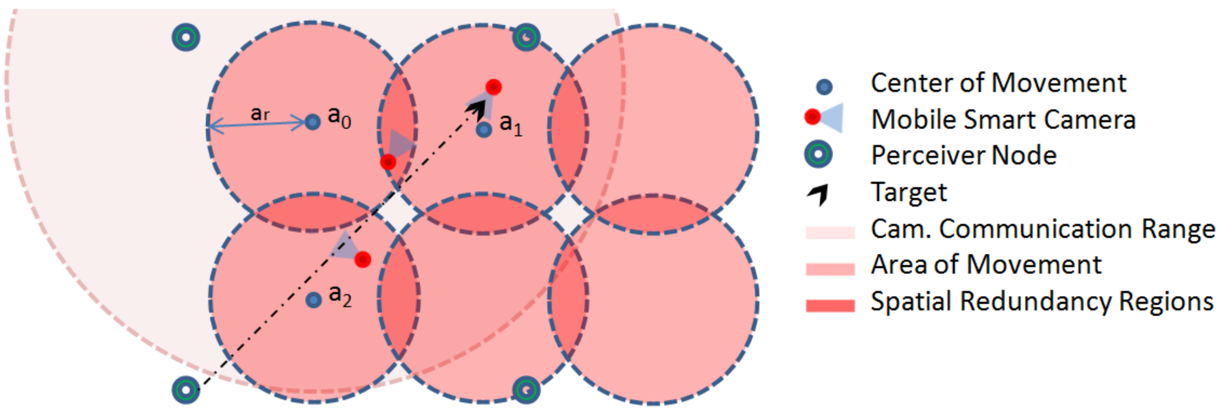

Figure 2: System Model

Our system model consists of the following components:

1. Targets

2. Perceivers

3. Active Cameras

The targets in our simulation represent people walking through the scenario with different trajectories and distributions. Each target has an unique identifier and the current position (x,y). They are placed on the workspace based on different distributions (i.e. random or Gaussian), and they update their position depending on their speed (1.5 m/s) and their movement. Once a target is sensed by one sensor, an identifier is assigned and a target request is generated.
The main objective of the perceiver nodes is to send the current location of targets in their range to the nearby cameras through targets requests.
The AC is an autonomous node containing a mobile entity (e.g., a quadrocopter as depicted in Figure 10), processing capabilities (CPU, memory, etc.), a camera and a communication interface.
In this thesis, the pose of the ACs is reconfigurable in terms of both position (x, y) and orientation. By changing the position on demand, ACs are able to meet the requirements for sophisticated computer vision. The mobile platform could be given in form of an unmanned vehicle, e.g. a mobile robot or an unmanned air vehicle as described in Introduction. The smart camera, need to contain a computing unit in order to carry out image analysis and handle the organization of the mobile platform as well as the cooperation between multiple active vision nodes.

### 2.2.1 Targets and Target Requests

In this thesis, targets are assumed to be added dynamically during runtime rather than given initially in order to make the system as independent as possible from a priori knowledge. The targets enter and leave the scenario continuously, but different placement distributions and trajectories can be set.

- Placement distributions:
  - Gaussian Placement; the new targets appear in the scenario dynamically during runtime following a Normal distribution, with mean and deviation as configuration parameters.

  - Random Placement; the new targets appear in the scenario dynamically during runtime following a random distribution, with mean and deviation as configuration parameters.
  - Simple Placement; the new targets appear in the scenario dynamically during runtime with x and y determined as configuration parameters.

- Target trajectories:
  - Lineal trajectories; the targets position is updated following a uniform rectilinear motion with x, y and orientation as configuration parameters.
  - Bend trajectories; the targets position is updated following a uniform circular motion with a center of the trajectory (a,b), the radius and the initial start orientation as configuration parameters.

A target request consists of the location $l_o$, a time of occurrence and the target's identifier. Once a target request is sent it takes a time until the next target request for the same target is sent. This is the system's sampling rate.

### 2.2.2 Perceiver Nodes

Perceiver nodes are low-cost sensors, such as motion detectors, being able to detect targets in their sensing range. We assume a simple sensor model that can calculate the position of targets in their range with an error. The wireless network used for perceiver-to-AC communication is an IEEE 802.11 network in ad-hoc mode. The perceivers communicate with ACs on a dedicated channel and send target requests for ToIs in their vicinity. Moreover, they are able to identify and quantify occurring perceivable events in their sensing range. The perceiver nodes are assumed to be three-dimensional proximity measurement sensors as described in [MN03] and thus able to deliver the ToI's position and time of occurrence with a specific frequency and accuracy. Various non-contact sensors are available, which could be used for this purpose. These generally are based on lasertriangulation, phase- or amplitude-modulation based electrooptical transducers or ultrasonic transducers [Web99]. A pair of calibrated CCD cameras can also be used in combination with stereo image-processing techniques to estimate the distance between a target and a center point between the cameras [DG07].

### 2.2.3 Surveillance Area and Virtual Nodes

We assume the surveillance area to be subdivided to some circular regions called Area of Movement (AoM) and we assign a AC to each region. These regions are the Area of Movement of the cameras, where the cameras can move around freely to track targets. They are determined by a Center of Movement (CoM) and a radius (ar). There can be overlapping regions between neighbouring AoM, so called “spatial redundancy regions”, which can be utilized by the cameras to cooperate to improve the system’s utility.

Virtual nodes make way for phenomenon adaptivity. While the cameras track events, the concept of virtual nodes allows for adapting the position of the AoM, trying to optimize the node’s position based on the previous distribution of targets that were detected. We assume that the movement of the AoM is restricted to two dimensions and by the camera’s radius as depicted in Figure 3. Thus, four possible neighbourhoods can be created, named A1, A2, A3 and A4 .
We assume that the virtual node needs information about its position and orientation in order to cooperate efficiently with other nodes. The initial configuration can be obtained by manual configuration at deployment or by camera calibration techniques. Nevertheless, we assume that each node is equipped by an appropriate positioning technology such as GPS in outdoor scenarios or by IEEE 802.11 WLAN positioning in indoor scenarios. The Euclidean distance between ACs and targets can be calculated.

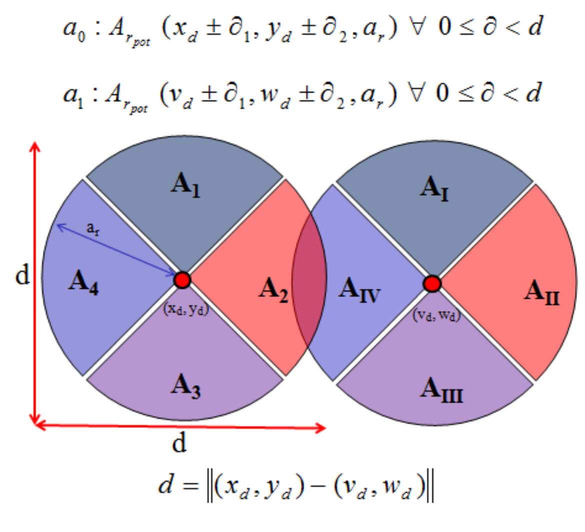
$a_0 : A_{r_{pot}} (x_d \pm \partial_1, y_d \pm \partial_2, a_r) \ \forall \ 0 \le \partial < d$
$a_1 : A_{r_{pot}} (v_d \pm \partial_1, w_d \pm \partial_2, a_r) \ \forall \ 0 \le \partial < d$
$A_1$
$A_I$
$a_r$
d
$A_4$
$A_2$
$A_{IV}$
$A_{II}$
$(x_d, y_d)$
$(v_d, w_d)$
$A_3$
$A_{III}$
d
$d = \|(x_d, y_d) - (v_d, w_d)\|$

Figure 3: AoM: Area of Movement

### 2.2.4 Communication Model

As seen in 2.1.3 a distributed ACN is a network of processors in which the nodes do not have direct knowledge of the state of other nodes. A node can retrieve the state of another node only by receiving a message from that node. Thus a communication between nodes has been implemented to make possible cooperation between the neighbouring nodes. Each AC is an autonomous agent capable of communicating with nearby nodes. Then to know how many neighbours are there, the cameras are sending Heartbeat MSG and listening for a Heartbeat MSG containing the id and the camera and node actual position. These way neighbourhoods are created and dissolved constantly. And the information of the nodes and cameras evolve with the scenario. Thus, Nodes can use this information to focus on different strategies like maximal area coverage or the creation of spatial redundancy regions.

```
Procedure generateHeartbeat();
begin
    Every x seconds broadcast a Heartbeat message including camera id, camera
    position and node position.;
end
Procedure processHeartbeats();
begin
    for every received Heartbeat do
        if neighbourhood cache contains information about neighbour already then
            Update neighbour information in cache.;
        end
        Add neighbour information to cache.;
        Set neighbour information time-to-live TTL = 2.5 seconds;
    end
    for every neighbour information i in cache do
        Decrease i TTL;
        if $i.TTL = 0$ then
            Remove i from cache;
        end
    end
end
```

**Algorithm 1:** Heartbeat Algorithm

# 3 Concept and Implementation

## Contents

In this chapter, we introduce the tools needed for the simulation of the distributed event-based protocol and present our concept.
The simulator we have used to simulate the distributed event-based protocol is the Multi-Agent Simulator Of Neighbourhoods (MASON) [SLB] : a fast, easily extensible, discrete-event multi-agent simulation toolkit in Java. MASON was designed to serve as the basis for a wide range of multiagent simulation tasks ranging from swarm robotics to machine learning to social complexity environments.
A number of researchers have built multi-robot systems in order to investigate the cooperation and pooled capabilities of distributed robots, focusing on team organization, interaction and task performance.
In our proposed approach the ACs can only travel in their area of movement. Thus, the goal is to find the optimal position of each AoM based on the number of events which occurred before.

## 3.1 Simulation Environment

MASON [SLB] is a single-process discrete-event simulation core and visualization library written in Java, designed to be flexible enough to be used for a wide range of simple simulations, but with a special emphasis on swarm multi-agent simulations of many of agents. The system is open-source and free, and is a joint effort of George Mason University's Computer Science Department and the George Mason University Center for Social Complexity.
One source of interest has come from social and biological models, notably ones in economics, land use, politics, and population dynamics (for example, [Axt01],[EA96],[GT05]). Another source comes from the swarm robotics community, particularly as homeland security and defense interests have bolstered investigations of large numbers of "UVs" (Unmanned Aerial Vehicles, Unmanned Underwater Vehicles, etc.) for collaborative target observation, reconnaissance, mapping, etc. [BJ00],[FP01],[Par02b],[Par03].
MASON is written in Java in order to take advantage of its portability, strict math and type definitions and object serialization. The toolkit is composed of three layers: the utility

layer, the model layer, and the visualization layer. The utility layer consists of classes which may be used for any purpose. These include a random number generator; data structures more efficient than those provided in the Java distribution; various GUI widgets; and movie and snapshot-generating facilities. Next comes the model layer, a small collection of classes consisting of a discrete event schedule, schedule utilities, and a variety of fields which hold objects and associate them with locations. This code alone is sufficient to write basic simulations running on the command line.
The visualization layer allows for GUI-based visualization and manipulation of the model. Figure 4 shows a simplified diagram relating basic objects in the model and visualization layers. For most elements in the model layer, down to individual fine-grained objects in the model, there is an equivalent "proxy" element in the visualization layer responsible for manipulating the model object, portraying it on-screen, and inspecting its contents. A bright line separates the model layer from the visualization layer, which allows us to treat the model as a self-contained entity.

### 3.1.1 The Model Layer

MASON's model layer does not depend on the visualization layer and can stand-alone. A MASON model is entirely contained within a single instance of a userdefined subclass of MASON's model class *SimState*. This instance contains a discrete event *Schedule*, a *MersenneTwister* random number generator, and zero or more fields.

MASON employs a specific usage of the term **agent**: a computational entity which may be scheduled to perform some action, and which can manipulate the environment. MASON does not schedule events that are sent to an agent; rather it schedules the agent itself to be stepped (pulsed or called) at some time in the future. Hence MASON's agents implement the *Steppable* interface.

MASON's **fields** relate arbitrary objects or values with locations in some notional space. Some of these fields are little more than wrappers for simple 2D or 3D arrays. Others provide sparse relationships. An object may exist in multiple fields at one time and, in some fields, the same object may appear multiple times.

### 3.1.2 The Visualization Layer

Objects in the visualization layer may examine model-layer objects only with the permission of a gatekeeper wrapper around the *SimState* called a *GUIState*. When running with a GUI, it is this class that is responsible for attaching the *SimState* to visualization (or detaching

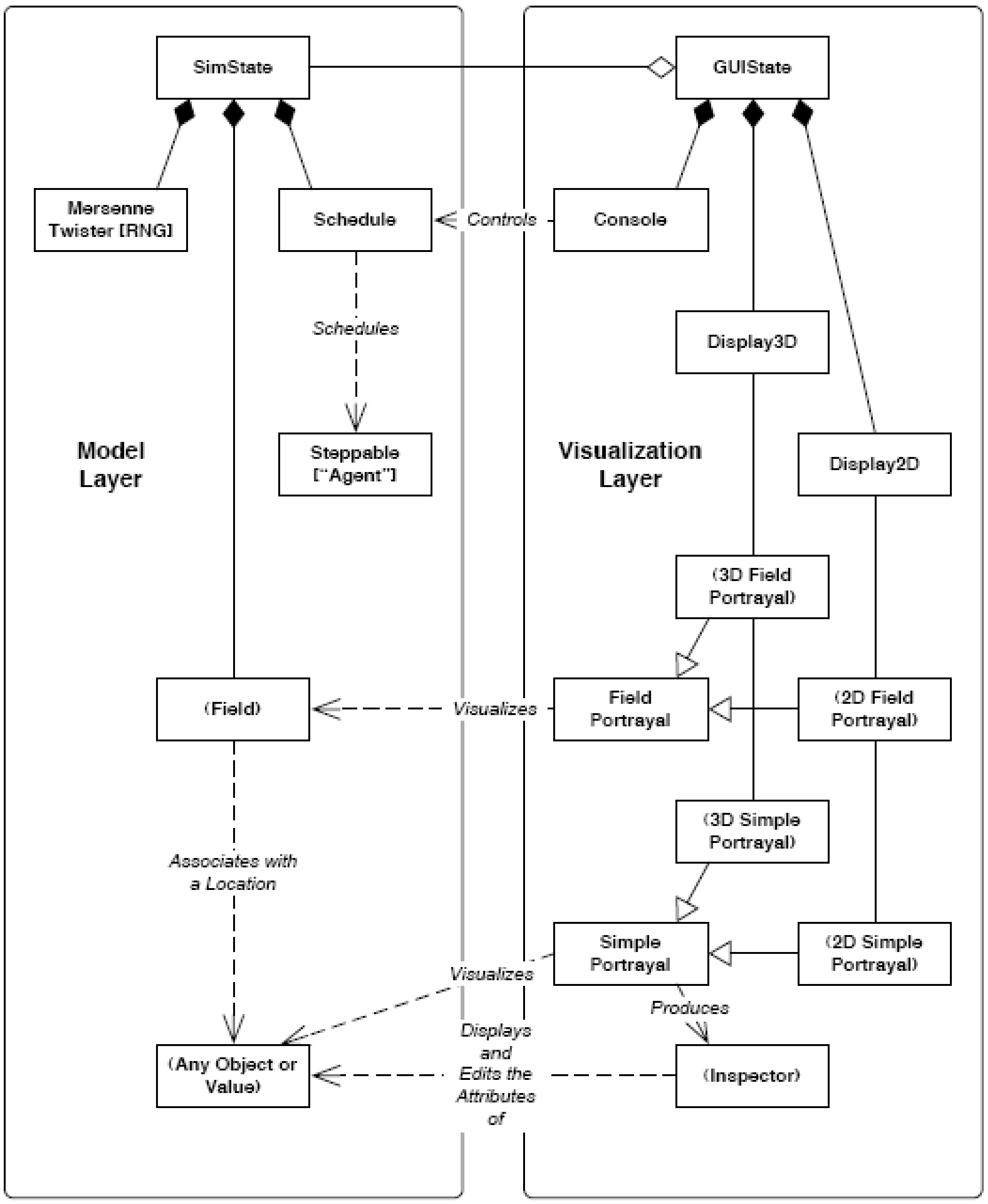

Figure 4: Model and Visualization Layers

it), and for checkpointing the *SimState* to or from the disk. As certain objects in the visualization world need to be scheduled (notably, windows need to be refreshed to reflect changes in the model), such elements may "schedule" themselves with the *GUIState* to be updated whenever the underlying *Schedule* is pulsed, but not be scheduled on the *Schedule* itself. This allows the visualization layer to be separated from the model.

Because of the separation of model from visualization, MASON models are usually created in two stages (refer again to Figure 1). First, the author develops the model proper as a self-contained subclass of *SimState*. After this, the code is completed, the MASON model should be able to run on the command line as a GUI-less application. Next, the author creates a *GUIState* to encapsulate the *SimState*, attaching portrayals and displays. At this point, the simulation can be visualized. The author can create further *GUIStates* to visualize the *SimState* in different ways.

### 3.1.3 Cooperative Target Observation with Unmanned Vehicles

In 2004 [SLP04] the MASON's team examined the effectiveness of various algorithms which manage mobile UAV agents (called observers) to collectively stay within an "observation range" of as many randomly moving targets as possible. Observers and targets are situated in a sparse continuous 2D or 3D field in MASON. This environment is used to examine "tunably decentralized" cooperative algorithms, whereas a parameter is utilized to gradually shift the algorithm from one global decision-making procedure to individual per-agent procedures.

### 3.1.4 Agent Communication

In MASON, communication protocols are not implemented. Thus, a protocol was needed and created under the following premises. The communication is based on a shared media model. This means that all AC and sensors have one or more network interfaces that are connected to a channel (air).

A channel represents a particular radio frequency with a particular modulation and coding scheme. Channels are orthogonal, that means that they are independent from each other, i.e. packets sent on one channel do not interfere with the transmission and reception of packets on another channel. The basic operation is as follows.
Every packet that is sent on a channel is received by all the elements connected to the same channel. In case a AC receives a packet, it first determines if it is possible for it to receive the packet. This is determined by the radio propagation model, based on a transmitter range.

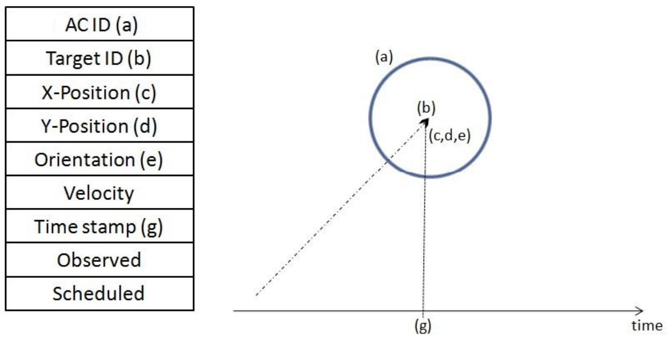

Figure 5: Target-Information-Messages

Perceivers communicate with the ACs on a separate communication channel in contrast to the AC-to-AC communication. Incoming target-information-messages (TIM) as depicted in Figure 5, e.g. target requests from perceiver nodes or target updates from neighbouring ACs, are handled by a separate thread running on each AC in order to add new target requests to the target cache or update them. The TIM messages include spatio-temporal information about the target (e.g. its position and speed, a time stamp and status parameters such as its observation state or if it has been scheduled for observation). The exchange of TIM messages allows the ACs to collaborate on target acquisition, e.g. to avoid that a target is scheduled for observation by two (neighbouring) ACs. TIM messages are rather small having a length of 30 Bytes each, since seven fields contain integer values and the last two ones booleans. Incoming messages are processed immediately on their arrival.

### 3.1.5 Simulation Parameters

A typical simulation under our project with MASON basically consists of generating a scenario file describing the following inputs:

- **Scenario**: width, height and seed factor (used by the random generator).
- **Perceivers**: quantity, interval of detection and error introduced by sensors.
- **Targets**: the movement pattern, the frequency on new generated targets and speed.
- **Mobile Smart Cameras**: quantity, initial overlap and breakdown probability.
- **Virtual Nodes**: memory of the system, maximal movement, and strategies.
- **Communication**: perceiver-to-AC and AC-to-AC package loss.

These inputs are used for the simulation and as a result from this, an output file is generated with the data of the experiment. Prior to the simulation, the parameters that are going to be traced during the simulation must be selected. The output file can then be scanned and analyzed for the various parameters that we want to measure that can be used as data for plots generation for example.

## 3.2 DRofACN: Dynamic Reconf. of Active Camera Networks

In this section we explain DRofACN protocol [MWH11], which is responsible for the movement of the AC in its area of movement and of the detection of Targets of Interest (ToIs). DRofACN is a distributed control algorithm for dynamic reconfiguration of cooperating ACs. The basic idea of DRofACN is that a set of ACs collaborates for acquiring close up views of targets in a surveillance area. The AC control is based on the output of perceiver nodes, i.e. generated target requests, estimating the position of salient targets, so-called targets of interest (ToIs), within the ACs' actuation ranges. The main goal is to acquire views for biometric purposes, such as face detection. There should be one capture of each ToI before it leaves the AC network's workspace.

DRofACN is based on a state-machine which will be described in detail in the following. Initially a timer is initialized and the AC is set to IDLE mode waiting for observation tasks. In case the timer expires, all target requests within the target cache are processed in order to find the next observation task, i.e. the most-promising target request for image acquisition. A timer is chosen in order to reduce CPU load by triggering the scheduling process in discrete time steps only. Thus, it allows for scalability in overload scenarios.
In case a target request is found exceeding a pre-defined imaging quality threshold, the AC changes to MOVING mode. In this mode, the AC informs neighbouring ACs about its intention to observe the target and starts to move towards the target in order to acquire close up views. After arriving the AC enters OBSERVATION mode in order to acquire high-quality imagery. Afterwards, it returns to IDLE mode.

## 3.3 Node Reconfiguration Advantages

The scenario presented in DRofACN is a subdivided area with smaller subregions. A camera is placed in each of these subregions, and they can observe targets that travel in these areas (Area of Movement of the camera). If we focus locally on the area of movement, the cooperation between cameras is minimal, only exist cooperation in overlapping regions between neighbours. In addition to this, the only interaction is to broadcast the next target to be detected to neighbouring AC.

- Surveillance area: 250m x 250m.

- Target generation rate: 6 targets per second.
- Target width path: 15m.
- Camera placement: 6x6 grid camera placement.

For the described scenario the AoM of the cameras has a diameter approx. of 40m. A straight target trajectory through the center of one of the rows means that each target must be detected by one of the six cameras until it leaves the surveillance area.
For the target generation rate of 6 targets/second, each camera in the row must detect a target each second. In addition to this, targets can become salient in every point of their trajectory. Thus, to achieve maximal coverage, each camera must estimate the position of salient targets within their actuation range and acquire views for biometric purposes of a target in less than a second.

This problem, can be solved increasing the quality and sophistication of the AC so it can be faster, increasing the number of cameras in the surveillance area, etc. But all this solutions has a higher cost than cooperation.
A number of robots working together can perform a task more quickly or return better results than the same number of robots working independently. The level of performance enhancement is highly dependent upon the task being performed as some tasks are highly sequential and are not easily or cannot be done in parallel.
Thus, a number of AC can be used to cooperatively accomplish tasks that are beyond the reach of a single AC. If the neighbouring ACs from the up and down rows help these six cameras, the performance of the system for this situation is improved by three. For this situation, there are 18 AC that cooperate to detect the six targets per second ratio. Thus, the time for detection is relaxed and the system can reduce the number of cameras on the system, use less sophisticates AC . . . and reduce the overall cost of the system
.

## 3.4 Node Event Utility Reconfiguration

In this section is explained the Node Event Utility Reconfiguration protocol which is responsible for managing the AC's Area of Movement (AoM). This protocol is based on a distributed control mechanism. The basic idea is to optimize the position of the AoM of the AC, in order to optimize DRofACN's performance for acquiring close up views of targets in the surveillance area. The pose decision is based on the output of perceiver nodes, i.e. generated target requests, estimating the position of salient targets, so-called targets of interest (ToIs), within the virtual node actuation ranges. The main goal is to optimize the position of all the nodes following different strategies in order to achieve the maximal number of ToIs detected.

This kind of adaptation, equips the AC network with some advantages. As we focus on the task of large area surveillance, multi-robot teams are ideal for such a task for the following reasons:

- A team of robots can continuously survey the entire area in parallel.
- Different types of robots may be sent to survey different types of areas.
- Teams of multiple, distributed robots may be more robust and fault-tolerant, as the loss of a single robot does not necessarily result in failure of the mission.
- Teams of small, inexpensive robots may be less expensive and less detectable than a single larger vehicle.

As seen in Section 1.3 the strategies that can follow an AC system are diverse and a lot of problems can be solved due to them. In our proposed approach, we have designed a protocol that is very useful in scenarios where people follow some kind of trajectories, something common in real-world scenarios.

The idea is to get a maximal coverage of the area, but giving crowded areas/trajectories a high priority. Thus, the nodes cooperate to track more crowded zones.
The ACs can move around their area of movement. The nodes must adapt the position of the AoM so the cameras are able to optimize the number of targets detected in their range by minimizing the time spent for travelling form one target to the subsequent one.

The protocol achieves this adaptation through a utility function with two inputs, the target and the neighbour cache. The adaptation process can be summarized as follows:

For each possible new position the camera computes the utility function. Once it has all possible destinations with the information of the number of targets in those positions, it computes the best position for adapting its AoM. Periodically the node updates its pose, but also storages info about the position of the past targets. Based on this, the nodes are able to find the optimal position.

In a real-world scenario e.g. the entrance of a main station, people enter and leave the surveillance area from different places. Thus, the trajectories are time-dependent.
E.g. the load of people arriving or leaving Hannover at 8:00 at the Main Station is not the same as at 12:00. Due to the utility function the pose of the nodes will evolve with the

system over time.

### 3.4.1 Problem formalization

In this section the problem is treated formally. The formalism for CMOMMT has been stated by Parker in [Par02a] initially:

1. S : two dimensional, bounded, enclosed region as area of interest.
2. V : team of m robot vehicles, $v_i$, i = 1,. . . ,m
3. $S_{v_i}$ : subregion of S assigned to the $v_i$
4. AoM coverage($v_i$): subset of $S_{v_i}$ observable by AC $v_i$. This region varies as the robot $v_i$ moves inside $S_{v_i}$.
5. A set of n targets, $o_j$ (t), j = 1,2,. . . ,n, such that target $o_j$ (t) is located within region S at time t.

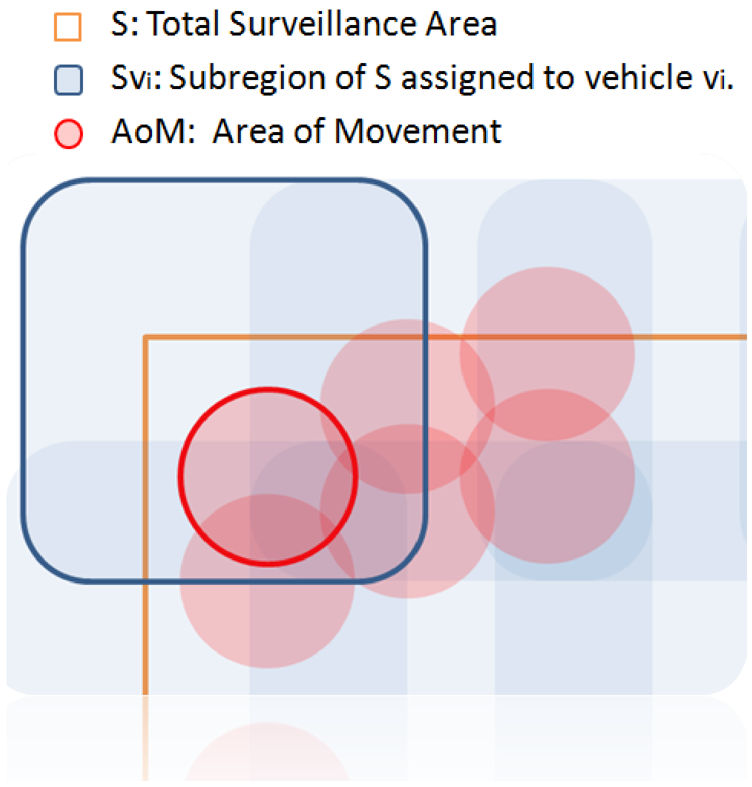
S: Total Surveillance Area
Svi: Subregion of S assigned to vehicle vi.
AoM: Area of Movement

Figure 6: Surveillance Area Description

The goal is to develop an algorithm that maximizes the visibility of all the targets in S. For this purpose, we stated the following requirements which have to be fulfilled by our protocol:

- The less targets a region of S has, the less AC spatial redundancy regions should be created in such a region.
- The more targets in a trajectory are detected in $S_{v_i}$ due to cooperation, the less spatial redundancy regions should be created in other regions to observe this trajectory.
- The more targets an AoM covers, the less time the AC requires to treat the TOIs so more targets can be detected and the more energy and resources can be saved.

### 3.4.2 Adaptation Process

In this section we explain the adaptation process in detail. The adaptative process requires the following data in order to perform the calculations: The target cache where all the data of the targets detected in the area is stored and the neighbour cache (data of the neighbouring nodes).

$A_{r_{pot}}$: This area covers all the possible positions of the center of the AoM. $(x_d \pm \delta_1, y_d \pm \delta_2, a_r) \forall 0 \leq \delta \leq d$. Where "d" is the maximal distance of movement for the center. Thus, if this value is very high the computational cost of checking all the values can increase quadratically. In order to overcome this problem, the area is discretized by a constant-sized grid to reduce the computational cost.

$$u'_e(p) = (1 - w) \cdot e(p) \cdot l^2(p) + w \cdot e(p) \cdot l(p) \forall p \in A_{r_{pot}}$$
$$u_e(p) \leftarrow M \cdot u_e(p) + u'_e(p)$$

**M**: This factor [0..1] controls the memory of the system, for fast evolving scenario a low value should be appropriate, for slow or not evolving scenarios a high value is more suitable. But for unknown scenarios a value of 0.5 should be used.

**e(p)**: This function returns all the targets in range of the virtual node, with the center of movement (CoM) at the point p.

**l(p) and l$^2$(p)**: These functions determine whether the targets detected at p are in redundancy regions or not, using the information of the neighbour cache.

**w**: This factor [0..1] determines the strategy for creating spatial redundancy regions. With a high value, the system will try to get maximal coverage, with low values the system will focus on the creation of spatial redundancy regions for cooperation.

Based on this function and sampling period T (collecting data about the number of targets and trajectories for each possible position), the number of targets in range of the node is measured, and the utility function is evaluated. As every part has been presented, a global and exact explanation is given in the next sections.

### 3.4.3 Initial Node Configuration

The virtual node in charge of optimizing the pose of the AoM needs some inputs to work with. Each node has the following parameters:

- Position: The initial configuration can be obtained by manual configuration at deployment or by camera calibration techniques. Nevertheless, we assume that each node is equipped by an appropriate positioning technology such as GPS in outdoor scenarios or by IEEE 802.11 WLAN positioning in indoor scenarios.

- W: As described above in Section 3.4.2, this factor defines the node's strategy. In this thesis all the nodes are initialized with the same strategy, but future works can study the possible improvements of adding nodes with different objectives i.e. so-called anti-agents.

- D: This factor determines the distance that the AoM can move in X-Y direction. In this thesis all nodes have the same value, but future work for example can focus on studying different strategies. The central nodes could have more freedom of movement to achieve most of detections on the borders of the area of surveillance.

- M: fractional learning rate of the node, each node stores its utility function, with this value they can remember trajectories or focus on discovering new ones.

- TTU: The Time To Update of the node is when the nodes calculates its utility function and computes the new pose of the area of movement.

```
1  initialization;
2  set timer;
3  set w;
4  set m;
5  foreach target information in Target Cache do
6      foreach position [i,j] in my actuation range do
7          if pos inRange(node) then
8              if pos inRange(Neighbour Cache) then
9                  $E[i][j] \leftarrow E[i][j]$+(1+w);
10             else
11                 $E[i][j] \leftarrow E[i][j]$+w);
12             end
13         end
14     end
15 end
```

**Algorithm 2:** Phenomenom Learning Phase

### 3.4.4 Phenomenon Learning Phase

In this section the target trajectory and distribution acquisition process are presented. The node will store all this info until TTU expires, then as presented in Section 3.4.5 the pose of the Area of Movement is updated.
Different strategies can be presented for this protocol, as we can see above. Due to this fact, the nodes must be first of all initialized with D, TTU, W and M ,see Algorithm 2 line 2.

To calculate the optimal pose through the utility function, e(p) must be acquired. This is carried out in a distributed manner as follows. Node i stores the number of all events detected (that are stored in the target cache, see Alg. 2 line 5) in range for all the locations within its movement range. Thus, two problems appear:

- High Computational Cost: Each node can move a distance of D in X and Y, that means that a matrix [D][D] is needed to store all the targets in range for each possible position and for each node. Thus, the computational cost increases with the number of nodes and with D quadratically.
  The solution presented discretizes the possible positions, taking just the number of events for N positions with a gap between them of D/N. This way, the systems always work with matrix E[N][N], where N is 20 in our simulations, reducing the computational cost.

- W Strategies: The utility function, has different possible strategies controlled by the W factor. Thus, when a ToI is in range of the node i (see Alg. Obs. line 7), it needs to know if this event is in range of the neighbours (through the Neighbour Cache) or not as depending on the strategy, it should focus on spatial redundancy regions or in maximal coverage.
  On the other hand, the nodes are updating their position to optimize the pose of the AoM. Thus, when a target becomes ToI, for all the position i,j in range, **E**[i][j] must be incremented by (1-w) if the target is actually in range with other nodes or by w if the node i is the only one that can cover it.

### 3.4.5 Pose Actualization Phase

Since only a subregion of $Sv_i$ is covered by the AoM of the $v_i$ and its neighbours, events that occur in uncovered parts of $Sv_i$ , are not detected and can go through the system. Thus, e(p) acquired locally need not be a consistent probability density function in a global sense, but it serves its purpose of providing higher weights in the utility function, to points where more events can be covered by the AoM. To prevent the count variables from overflowing, the **E**(p) data is periodically aged as follows. After every time duration TTU (see Alg. 3 line 1), the node i scales the **E**(p) at each pose in $Sv_i$ by M. The aging step is:

$$\text{U(p)} \leftarrow M * U(p) + E(p)$$

Where the temporary variable E(p) stores the number of events since the previous aging step. This aging procedure gives a higher weight to more recent events than older events by degradation factor M each TTU (see Alg. 3 line 3). Thus, the phenomenon distribution learning continues every TTU. The update of U(p) makes possible the node reconfiguration.

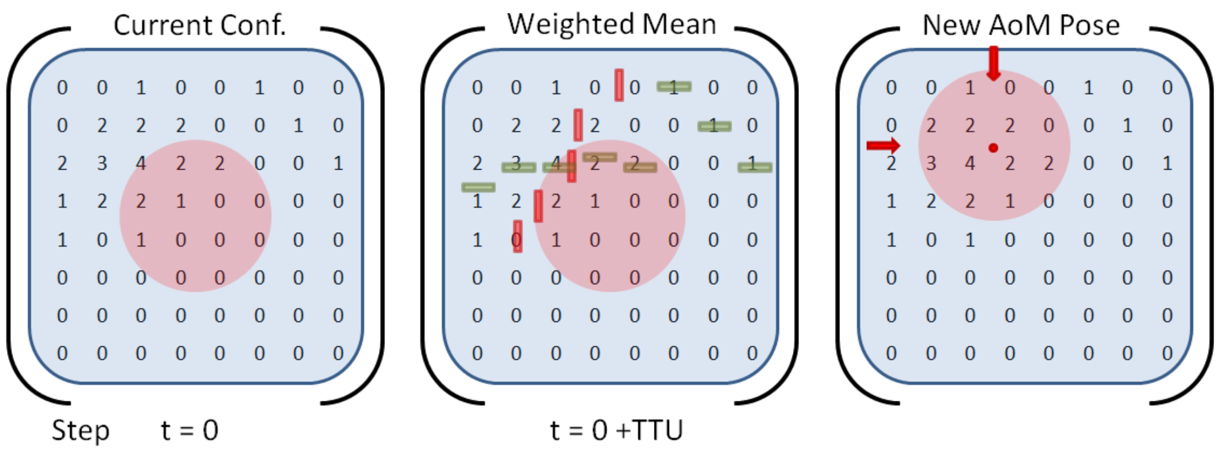

Figure 7: Pose Detection

At this point the U(p) of the node i has been updated and the node computes the optimal pose for its AoM. To achieve that, as we see in Figure. 7, the weighted arithmetic mean is computed for each row and column of the matrix **E**. Thereby, we get the index of the matrix where the most events occur. Thus, we achieve the optimal position for each row and column. The next step is to determine the best value for X and Y. The node measures which is the mean of the optimal positions calculated before. Then the node moves to this position and starts over with updating its utility matrix.

```
on timerExpire do
    foreach position [i,j] in my actuation range do
        $U[i][j] \leftarrow E[i][j]$+m*$U[i][j]$;
        $E[i][j] \leftarrow 0$;
    end
    foreach row $U[i][j]$ do
        $X[i] \leftarrow$ weighted arithmetic mean($U[i][j]$);
    end
    foreach column $U[i][j]$ do
        $Y[j] \leftarrow$ weighted arithmetic mean($U[i][j]$);
    end
    mySPM(positionX) ← mean X;
    mySPM(positionY) ← mean Y;
    set timer;
endon
```

**Algorithm 3:** Pose Actualization Phase

# 4 Simulation Study

## Contents

In the DRofACN [MWH11] several questions have been answered in terms of the local camera reconfiguration, e.g. how many resources, i.e. cameras, are needed in relation to the number of ToI or how long does it takes to detect a target after becoming salient. All of these assumed trajectories are placed with maximal spread over the scenario (see below for the definition of spread).
Nevertheless, people's trajectories vary over time and the spread is restricted in terms of its size, e.g. sidewalks, streets or delimited ways in concerts or airports. Due to this reason, experiments have been conducted to investigate how DRofACN works for scenarios with different spreads, and if the adaptation process proposed in this thesis really can improve the system's performance.
The experiments conducted are based on a prototype of an AC and its simulation. In Section 4.3, we describe the simulation of an AC network.

## 4.1 Experimental Setup and Simulator Parameters

In this section, the parameters and the scenario setup are presented. The area of the scenario to be observed is 250 x 250 units square. This is a common size for public places, e.g. such as the front yard of the Hannover Main Station as depicted in Figure 8. In the experiment, to compare our Organic Node Reconfiguration protocol with the DRofACN protocol, AC and the nodes are placed in the scenario following a grid pattern with a minimal overlap between the area of the neighbouring nodes (AoM of the AC). The AoM also is reduced as the number of cameras is increased in the system. That makes our protocol more precise but also limits the movement of the nodes trough the scenario.

1. **Number of cameras**: The system size is changed between 2x2 . . . 6x6 ACs. The actuation radius of each camera is calculated as follows:

$$ar = \frac{1}{2} \cdot \frac{250m}{camsInRow}$$

As explained before, the actuation range decreases with the number of ACs in order to support dynamic reconfiguration. Nevertheless, the number of spatial conflicts and communication per area is increased.

2. **Spread of trajectory**: Since we are interested in the system's behaviour in overload conditions, we set the target generation rate to 6 targets/s and the ToI rate to one.

   - Target generation rate: In our experiments the targets entering the surveillance area are generated with a specific target generation rate that determines the number of targets per second.
   - Target salient rate: All targets traversing are not targets of interest. The target salient rate was created to control the number of ToIs in the scenario. A target rate of one means that all targets within the system become salient.

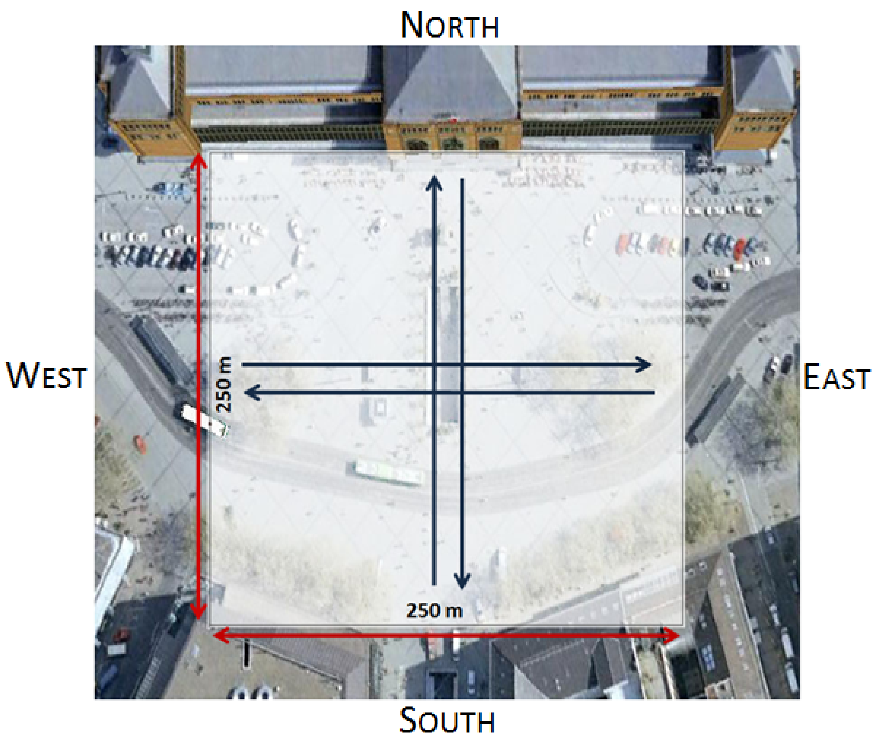

Figure 8: Hannover Main Station

The spatial density of the trajectory is defined by its spread. There is a difference, for example, between a target generation rate of 6 targets per second distributed over trajectories of 250m or 2,5m. As the spread of the target distribution in the trajectory is reduced, less cameras are available to detect the same number of targets as the density of targets in the trajectory increases. The trajectories which follow the targets entering the scenario in our experiment are straight-line trajectories that came from east, west, north and south as depicted in Figure 8.

The spread of the trajectories varies between 2.5m up to 250m in steps of 12m in order to investigate how the system evolves with the different trajectories and different ACs deployments. Figure 9 depicts different spreads.

Another important aspect is that since the ToIs move with a speed of 1.5 m/s, they need for example approx. 166s traversing the surveillance area in case of a straight-line trajectory. That means that for the 6 target/s rate there are about thousand targets in steady state that become ToIs following an uniform distribution for all the path through the surveillance area. Thus, some targets become salient at the end of the trajectory and there is no time for the ACs to detect them.

3. **Time dependent trajectories**: In a real-world scenario trajectories under a surveillance area can be static or they can evolve over time. We have investigated two different experiments, one with time-invariant trajectories with variant spread and another one with time-variant trajectories and static spread.

4. **Memory**: the node aging factor, has been set to 0.5 in order not to overflow the utility function with the old knowledge.

5. **Time To Update (TTU)**: The TTU parameter has been set to 50s plus a different random value of 50s for each node. Thus, the nodes can take their decisions knowing where their neighbours are.

6. **Node distance**: The distance each node can move from its center of movement is $1.95\cdot$ times the range's radius. The nodes can update their pose approx. to the center of the neighbouring nodes.

7. **W strategies**: The experiment focus on the creation of spatial redundancy regions, the w value has been set to a very low value of 0.1 giving high priority to cooperation and less priority to maximal coverage.

## 4.2 Performance Metrics

Several metrics have been applied to measure the performance of this protocol; TAR (target acquisition ratio), RT(reaction time), IDLE time, image quality, detections per target…

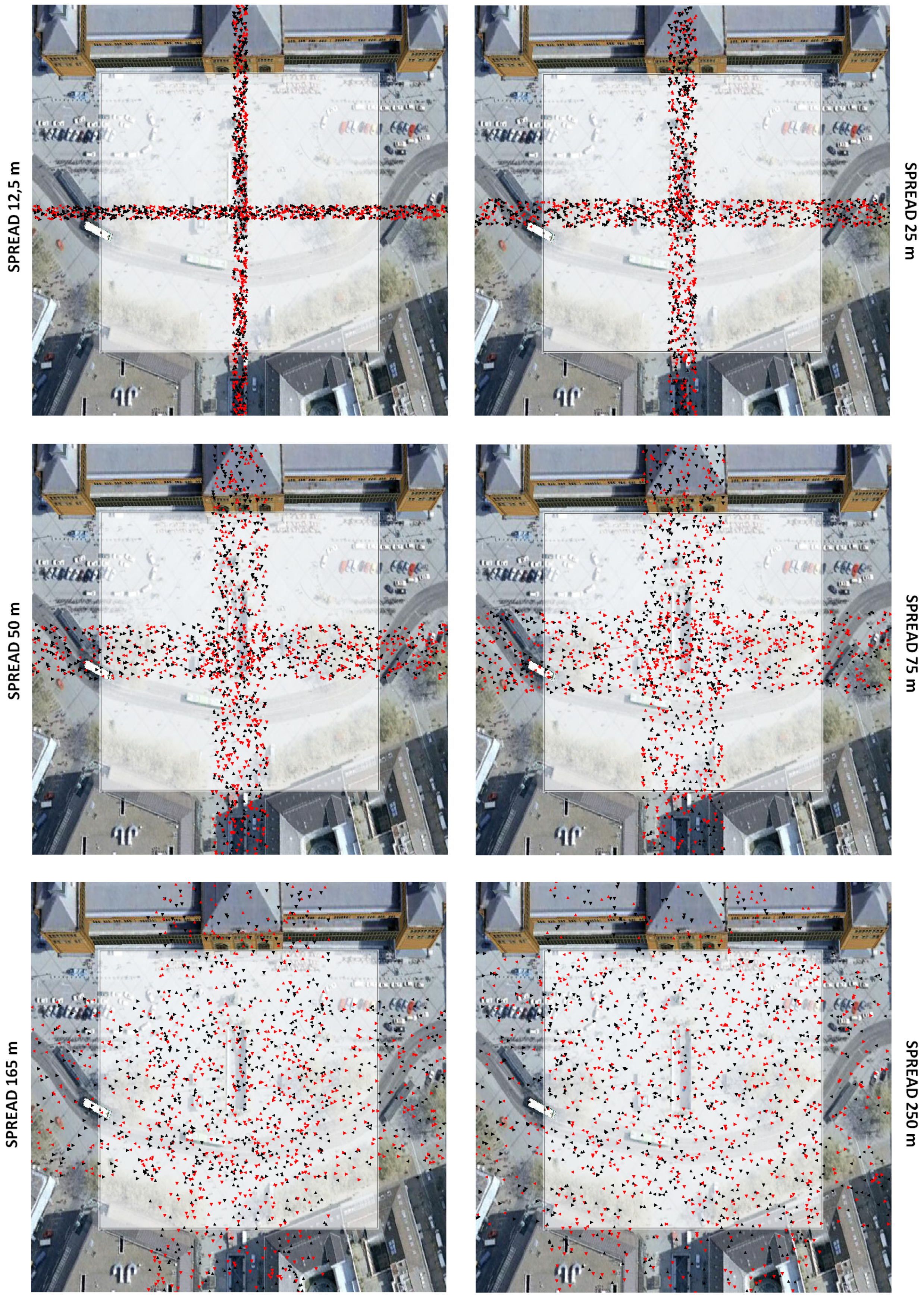

Figure 9: Trajectories with different spreads

- **Target Acquisition Rate (TAR)**: This ratio express the number of targets that become ToI in the area under surveillance, and are successfully observed before leaving, to the total of ToI entering the surveillance area during the experiment (T sec.). Thus, a TAR of one means that every target entering the surveillance area that has become a Target of Interest(ToI) is observed at least once before leaving.

$$TAR = \frac{1}{n}\sum^{T}\sum ToI_i \cdot \text{ observed(t)}$$

- **Target detection time or Reaction Time (RT)**: This metric measures how much time is needed to acquire a picture of a target after becoming a ToI. With this ratio, it can be studied how far the cameras in a selected scenario must travel. The protocol focus on the creation of Spatial Redundancy Regions to decrease this reaction time and then achieve a higher value of TAR.

- **IDLE time**: This ratio presents how much time are the cameras IDLE, during the experiment. A high value of this ratio means that there are too many resources (AC) which are not being used. If the TAR gives good results, the number of AC can be reduced. Bad results of TAR means more cooperation between ACs is needed.

- **Number of captures per target**: Each target should be observed at least once. This metric is important to investigate how far the observation condition is fulfilled in terms of disturbances e.g. network partitioning.

## 4.3 Simulation of an AC network

The AC node we propose relies on an off-the-shelf image sensor connected to an ARM processor. Using an ARM based processor allows for low power usage and mobility. Thus, it can be attached to a quadrocopter making way for reconfiguration in terms of the position and orientation. In addition to this, utilizing a standard processor allows for the usage of common software, as for example Linux and OpenCV, a collection of algorithms for computer vision. We decided not to build a proprietary hardware platform since our focus is mainly on the networked system architecture rather than on robot navigation techniques. The ACs used run Linux as operating system since it offers highly flexible and reliable networking capabilities. Since the presented algorithm is a lightweight algorithm it can be implemented on other resource-constrained platforms (e.g. Intel Atom chipset) without much effort, too. The wireless network used for inter-AC and perceiver-to-AC communication is an IEEE 802.11 network in ad-hoc mode. The perceivers communicate with ACs on a dedicated channel and send target requests for ToIs in their vicinity. Moreover, they are able to identify and quantify occurring perceivable events in their sensing range.

The perceiver nodes are assumed to be three-dimensional proximity measurement sensors as described in [MN03] and thus able to deliver the ToI's position and time of occurrence with a specific frequency and accuracy. Various non-contact sensors are available, which could be used for this purpose. These generally are based on laser triangulation, phase- or amplitude-modulation based electrooptical transducers or ultrasonic transducers [Web99]. A pair of calibrated CCD cameras can also be used in combination with stereo image-processing techniques to estimate the distance between a target and a center point between the cameras [DG07].

ACs investigated in this thesis are assumed to be mobile in x- and y-position and form a MANet. The ACs are installed at different heights, i.e. z-positions, in order to avoid collisions during reconfiguration in case of overlapping actuation ranges. The actuation ranges overlap in a way that every part of the entire surveillance area is covered by at least one AC. For the simulation of large AC networks, the discrete event multi-agent simulator MASON (Multi-Agent Simulator Of Neighborhoods) is used (Section 3.1).

In order to reach realistic simulation results, a IEEE 802.11-based wireless ad-hoc network as described in [OW10] has been implemented for communication with a transmission range up to 140m and a worst-case latency of 100ms. Since we assume the AC network to be in operation outdoor and only sparse radio obstructions to be present on the surveillance area, this communication range is common for today's wireless adapters in environments with sparse communication obstacles.

Figure 10: AR.Drone

Additionally, the physical behaviour of ACs in terms of moving and rotating speed has been modelled with linear equations respectively provided in the AC's datasheet. This type of quadrocopter moves with a speed of $5\frac{m}{s}$ , i.e. $18\frac{Km}{h}$. The flying time is about 12 minutes and the frontal camera is able to capture images of 640x480 pixels.
For modelling ToIs, persons are assumed to have a moving speed of $1.5\frac{m}{s}$. These targets are detected by perceiver nodes, which are equally distributed on the surveillance area to ensure

sampling the entire target trajectory of ToIs.
The observation time, i.e. the time needed by the image processing algorithm such as face detection, is assumed to be 1000ms. This is a reasonable value for an AC equipped with an ARM-chipset.
Simulating a scenario requires several configurations to be taken out. Since the exchange of TIM messages and target requests from perceiver nodes requires time stamps, the ACs and perceiver nodes are synchronized in terms of time by traditional time synchronization methods such as NTP. Target requests are send with a frequency of 1 Hz.
While simulating, each AC generates a log-file storing all data relevant for the succeeding analysis. Several experimental scenarios have been set up in order to measure and evaluate the performance for different trajectories' spreads of the presented protocol vs DRofACN results.

## 4.4 Trajectory VS Camera Placement

Real-world trajectories may change over time in terms of load and direction. This means that the nodes must adapt themselves to different kinds of situations all over the scenario. Some of these situations may yield better results than others. For example, a trajectory with small spread may pass a node at its center or at one of its borders. As the Organic Reconfiguration protocol focus on the creation of spatial redundancy regions between neighbouring nodes, the number of neighbours is important.

To understand the results of the experiment, it is important to know the relationship between the straight trajectories of the scenario and the node grid placement.
As the targets travel in straight trajectories following a uniform distribution centered in the middle of the scenario with a variable spread, the system results are affected because of the number of cameras and the grid placement giving better results for placements centered in the middle.

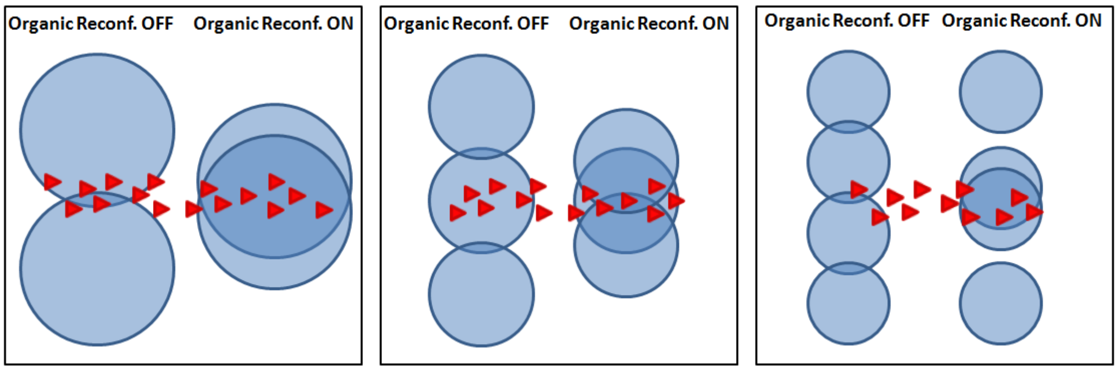

Figure 11: Trajectories VS Grid placement

As depicted in Figure 11, it can be achieved two different type of improvements.

- Optimal pose: The nodes update their poses and the target detection time is reduced. This improvement is present in all situations.
- Maximal cooperation: When targets traverse the node's center, the number of neighbours that can cooperate to detect targets is increased. This situation is depicted in the 1x3 placement in Figure 11.

Thus, the experiment presents a greater difference between results with the node reconfiguration ON and OFF, for odd grid placements like 3x3 or 5x5 than for even distributions like 2x2 or 4x4. When a surveillance area is designed, normally most of the trajectories will go through the center or near it. Thus, to increase the overall performance, the center nodes must have as many neighbours as possible. That is the case of a 3x3 placement. There is a node centered in the middle of the scenario with the maximal number of neighbours, eight; and they can change their pose to get every trajectory in the surveillance area.

## 4.5 Time-invariant Trajectories (with varying spread)

The results of the experiment for different node placements and spreads, depicted in Figures 12 and 13, show that the target acquisition ratio (TAR) can be improved significantly in case of trajectories with small spreads.

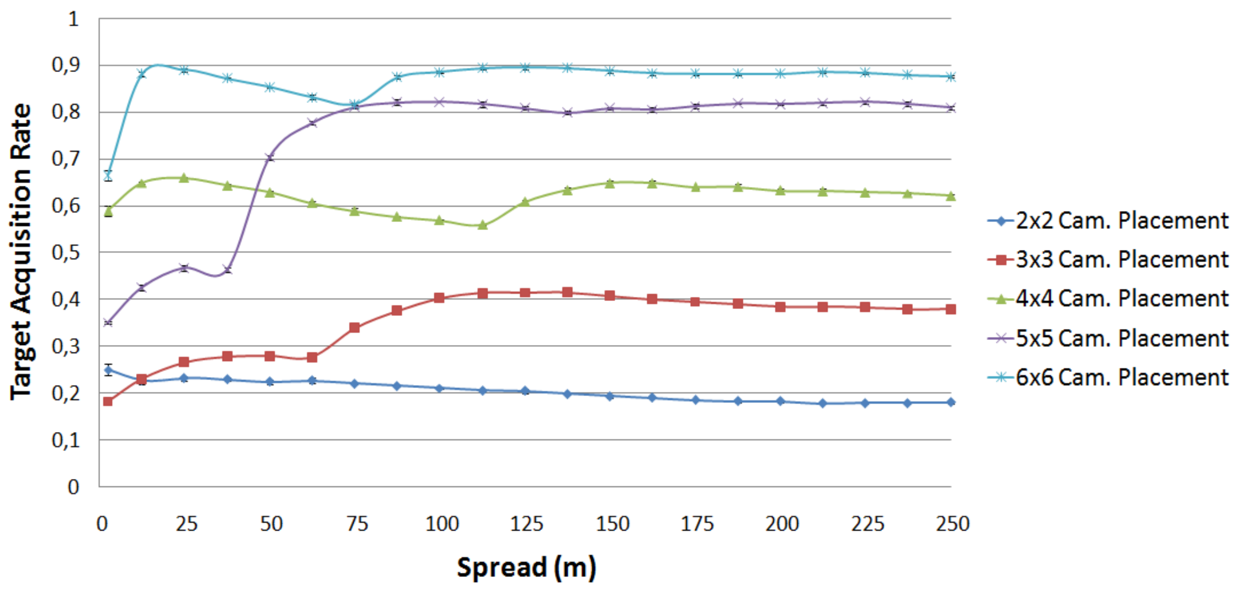

Figure 12: TAR without reconfiguration

The improvement for common path widths of trajectories induced by humans e.g. 2.5m up to 12 m is much higher than for big spreads i.e. 100m up to 250m. The explanation of this behaviour and other results are presented in this section, first in an overall view and later in more detail.

Scalability with respect to the target generation rate is an important criterion for AC networks. But also the spread of the trajectories affect the system. A high generation rate with a small spread means more dense trajectories and the need of more AC cooperation in this zones. Scalability concerns both the quality of the solution found to the wide-area target detection problem (i.e. resulting in a high TAR and low target detection times) as well as the number of resources needed. Results can be found in Figures 12,13,14 and 15. These Figures depict the relation of resources needed, i.e. ACs, to the algorithm's quality of service in terms of the number of targets observed (TAR) and the mean target detection time for a constant target generation rate but a varying spread.

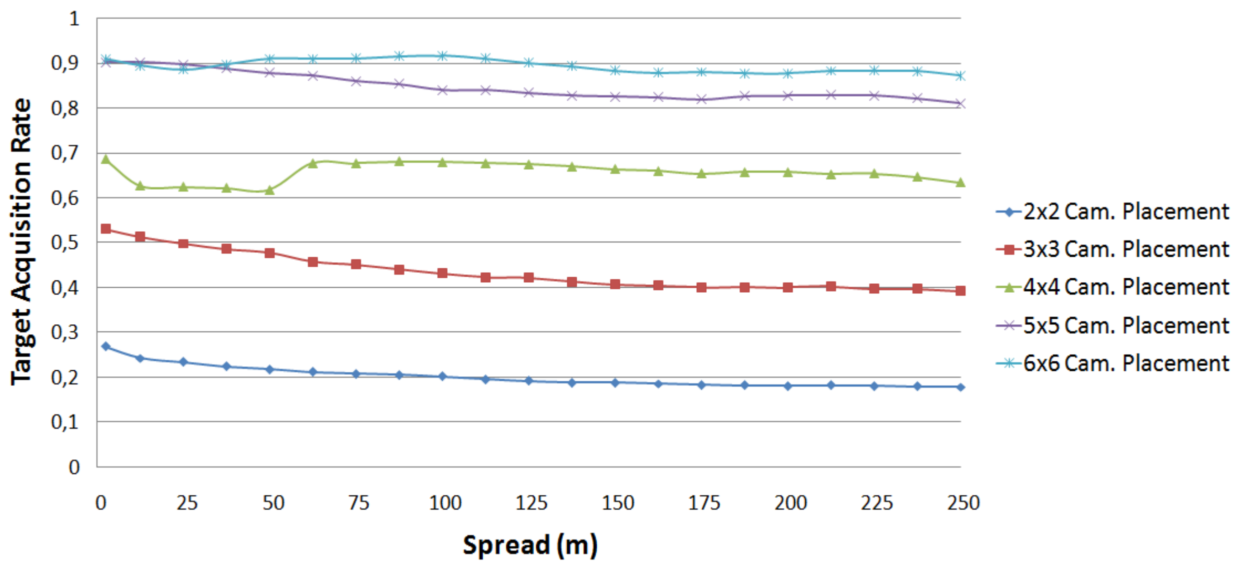

Figure 13: TAR with reconfiguration

By decreasing the spread of trajectories, the density of targets in the path is increased. This is due to the fact that the same number of targets is distributed over less area. Despite this fact, the adaptation of the nodes present better TAR results for spreads smaller than 75m that for the static nodes with maximal spread.

As the spread of the trajectories decreases, the nodes must update their position to get the optimal pose generating new spatial redundancy regions with their neighbours. These redundancy zones make way for cooperation between nodes and increase the target acquisition ratio.

On the other hand, for widths over 75m, the scenario is overflowed with targets, and the nodes do not need to reconfigure their pose to find the optimal pose. For these values, the improvement achieved by our protocol is small. But this small improvement is driven by the fact that some nodes can find optimal poses where more targets can be detected. In this case, as spread increases, cooperation decreases.

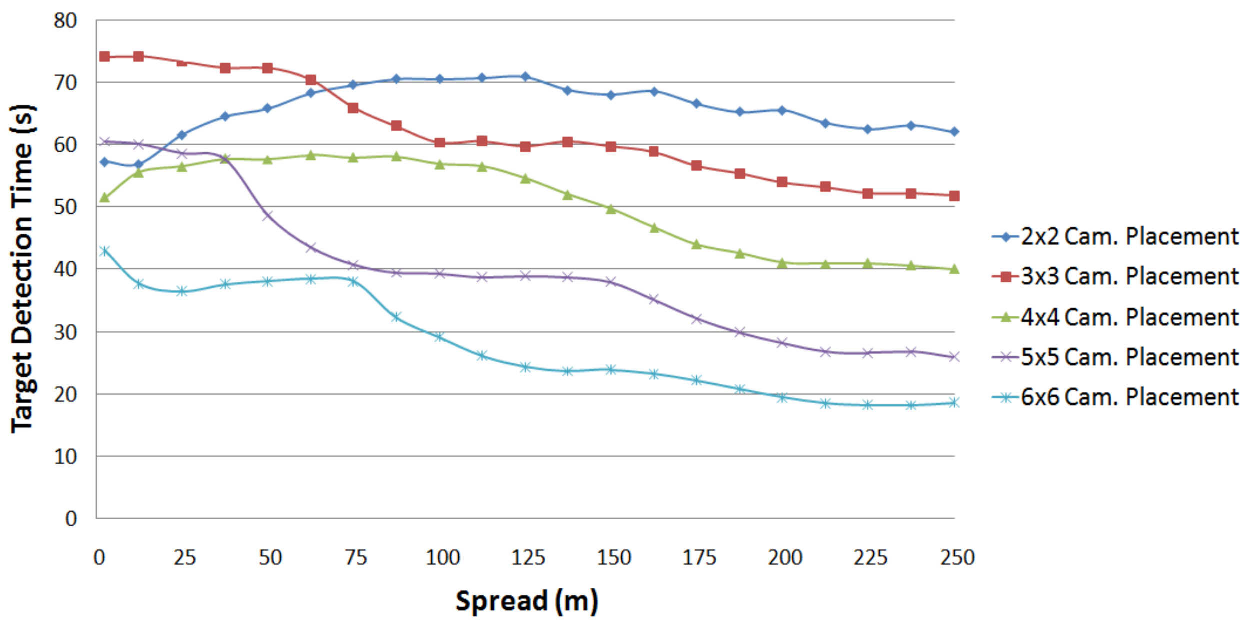

Figure 14: Target Detection Time without reconfiguration

Additionally, the different situations and the relation between spread and trajectories explained above can be found in the TAR results, as the improvement for camera networks of a size of 3x3 and 5x5 cameras is higher in comparison to networks consisting of 2x2 and 4x4 cameras. For camera networks of 3x3 and 5x5 cameras, the node placement is centered in the surveillance area. Thus, the ACs in the center row, have more neighbouring cameras to cooperate with (as explained in Section 4.4).

Figures 12 and 13 present the TAR results for the different camera networks without reconfiguration. All the networks have a TAR under 0.8 for small spreads. Due to the node reconfiguration, the system can achieve a TAR up to 0.8 with 25 ACs. Thus, the system can make an optimal use of the global resources improving the behaviour of the system.

Results presented in Figures 14 and 15 show how the reaction time is reduced for each configuration, and in a more intense way for odd node placements. As explained before, in these situations, there is more cooperation between nodes and then there are more cameras in the same area, what means that also in overload situations, i.e. for 2x2 nodes, more targets are detected, and the cameras can focus on ToIs that are in their regions. The cameras can create redundancy regions to balance the load and focus on nearby salient targets only.

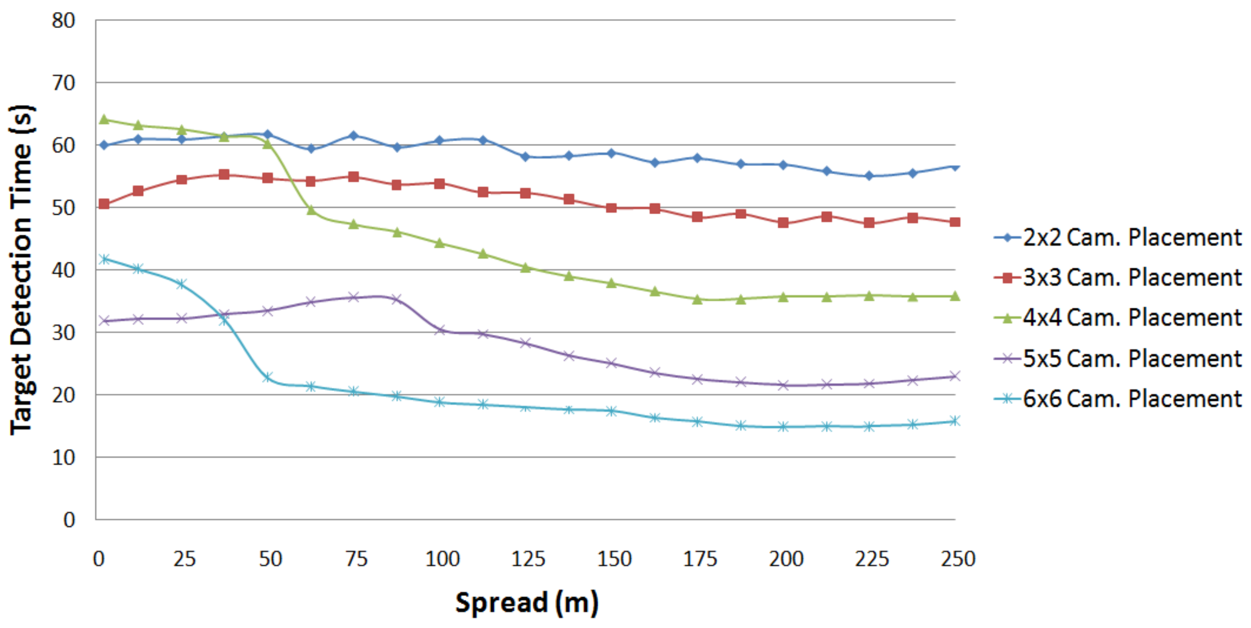

Figure 15: Target Detection Time with reconfiguration

The improvement for trajectories with high spread values is smaller, as the nodes are not able to cooperate by creating dominant spatial redundancy regions and they must focus on less dense trajectories. Thus, the cameras need more time to reach the targets.
The simulations have shown that the creation of spatial redundancy regions and the cooperation between nodes decreases clearly the target detection time. Thus, the cameras can detect more events and the TAR is increased achieving better results with less cameras.
A more detailed view of the behaviour of the scenario for the different camera networks is explained above.

This 2x2 camera network presents an overload scenario with 1000 targets in steady state and only 4 cameras to detect them all (see Figure 18). Thus, the results show that although the reaction time has been decreased, the TAR value cannot be increased. This is due to the fact that the cameras are in an overload situation and are constantly detecting ToIs. Thus, reconfiguration is less relevant for the overall improvement of the system performance.

- TAR behaviour (see Figure 16): In the first area depicted in the TAR results, the TAR value increases only a 7.2%. While in the other area, cooperation results are almost the same as the network without reconfiguration.

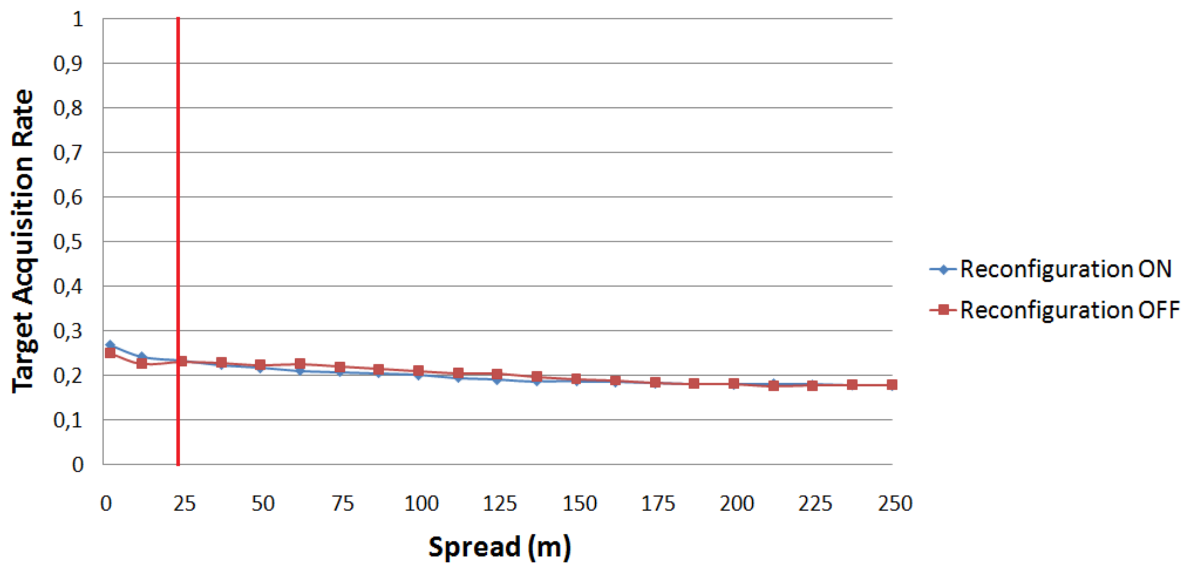

Figure 16: TAR 2x2 Camera Network

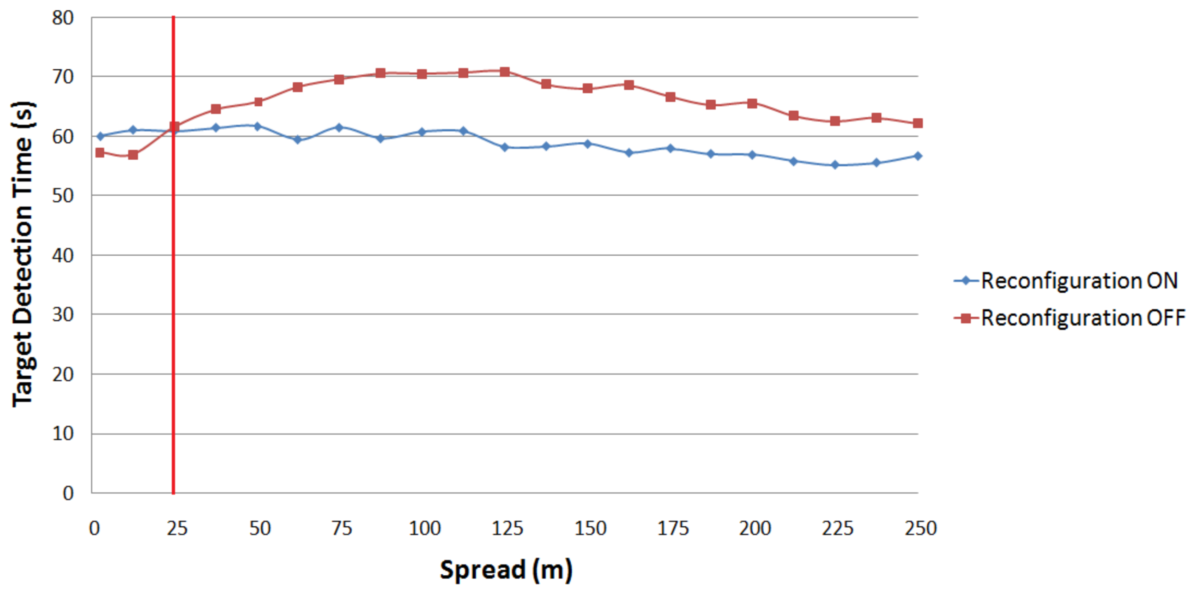

Figure 17: Reaction Time 2x2 Camera Network

- RT behaviour (see Figure 17): For small trajectories, all the cameras are observing targets. But they can observe just the targets in their AoM. Thus, there is a strange situation in the first area. The reaction time is increased for the network with reconfiguration, but the TAR values for this spreads are increased. This is due to the fact that cameras have updated their pose. Thus, they can observe targets that without node cooperation where too far. And then the target detection ratio increases. For the rest of spreads, the reaction time is reduced as expected.

As there are not enough cameras to detect all targets, the number of ToIs in the center of the scenario is clearly higher than in the borders and the trajectories converge there. Thus, as presented in Figure 18 the nodes tend to reconfigure their position towards the center to get the maximal number of undetected targets. But, what happens if the number of neighbouring nodes is increased? In that case, the results would be different as the cameras would cooperate inside their AoM to get all the targets, that are completely covered by the four nodes. The problem of this 2x2 configuration is that each camera has only three neighbours. Thus, the potential for improvement is restricted, as it focuses on the cooperation between neighbouring nodes.

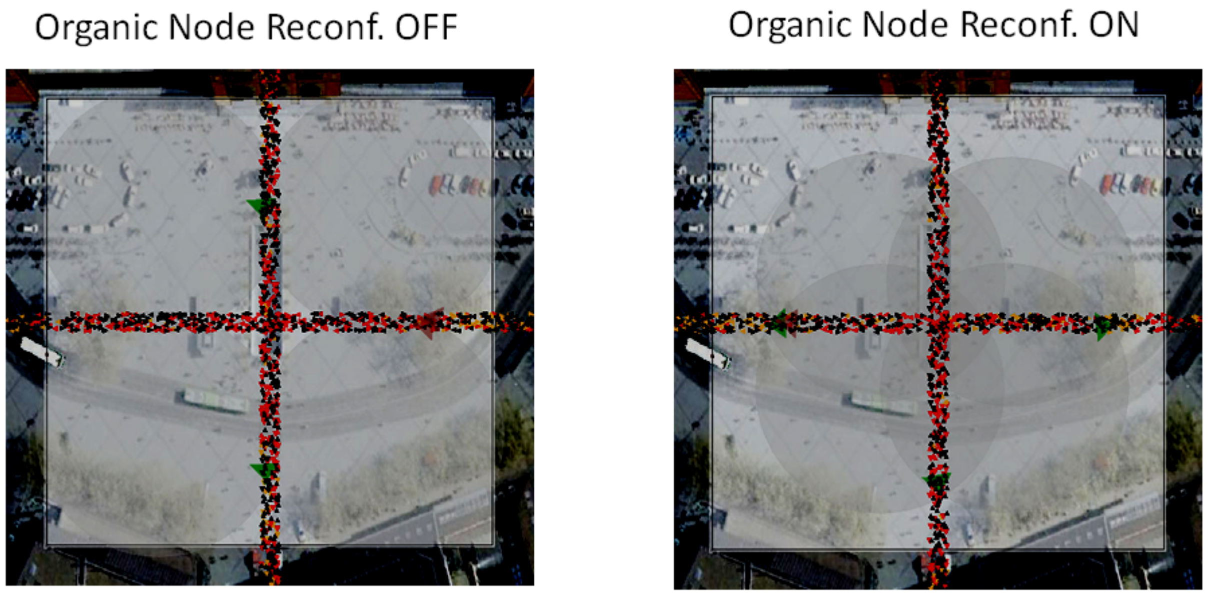

Figure 18: 2x2 Organic Node Configuration

Therefore, we increased the number of cameras up to nine. This configuration present the greatest improvements for our crowded scenario. The TAR is increased a 190 % and the reaction time is decreased a 33 % of the original values for the small spreads due to node reconfiguration. All the cameras within the scenario cooperate to detect targets, for all kind of spreads.

- TAR behaviour (see Figure 19): In a 3x3 placement, the camera in the center has eight neighbouring nodes. Thus, as explained in Section 2.2.3, the node can cooperate with all of them getting at the end the best pose for the scenario. Due to this fact, for spreads where cooperation is higher (spreads under 75m) the TAR results increase a 190%. For bigger spreads, node reconfiguration decreases.

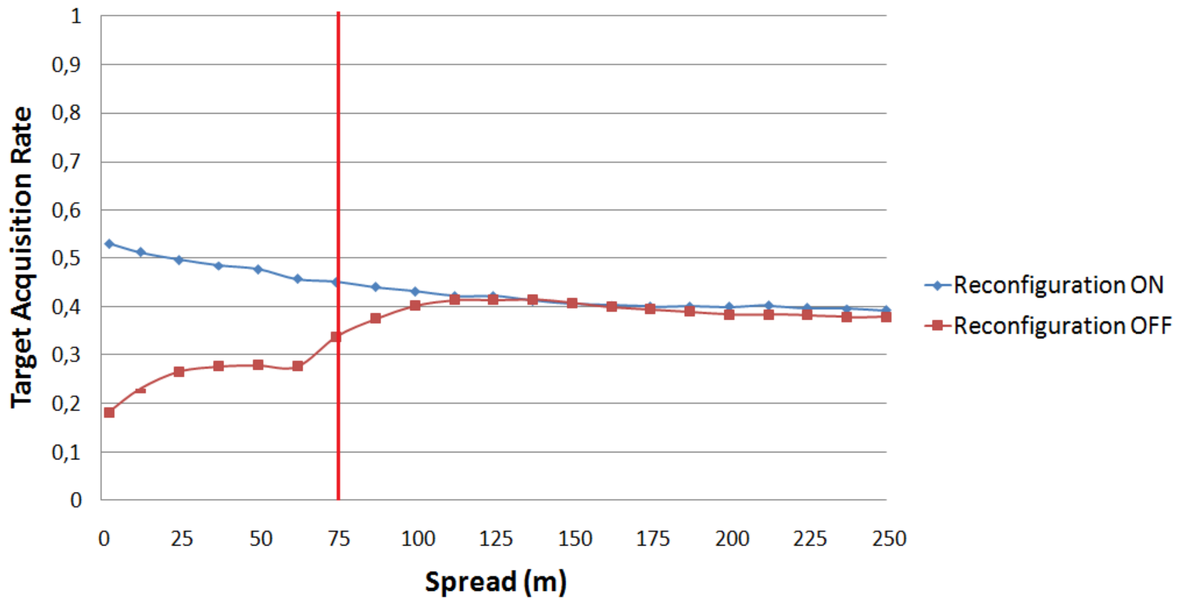

Figure 19: TAR 3x3 Camera Network

- RT behaviour (see Figure 20) : As explained above, the target detection rate results show that with more reconfiguration the overall system's performance increases.

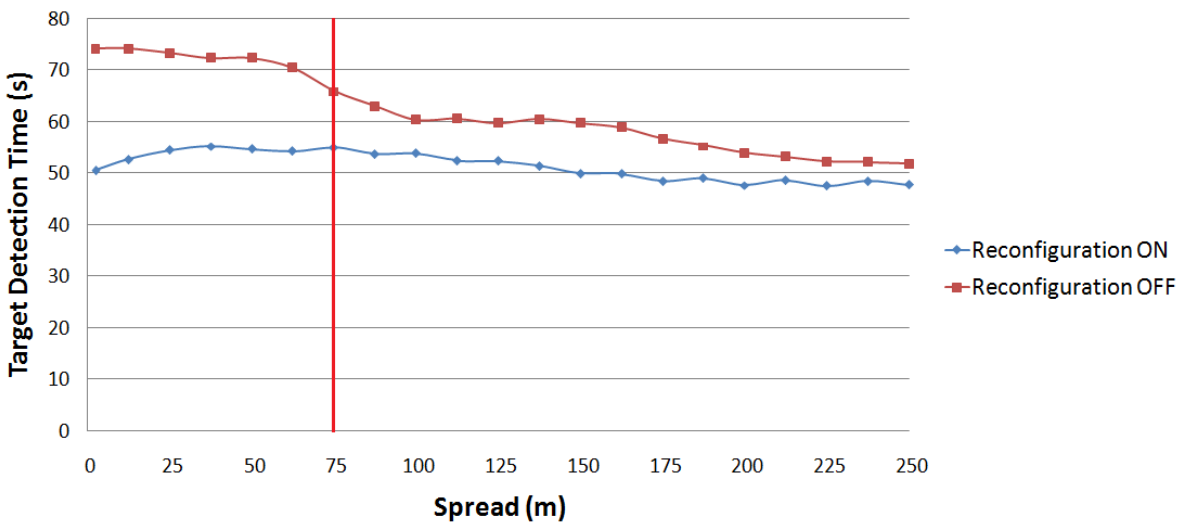

Figure 20: Reaction Time 3x3 Camera Network

The results are so impressive because with this configuration the protocol acts with all its power. In this configuration the AoM has a radius of approx 40m. Thus the scenario is divided into 9 different regions, that can adapt to trajectories perfectly.

In summary, the nodes update their poses in order to create spatial redundancy regions towards the center of the scenario due to the higher number of undetected targets there (see Figure 21). In case of nine cameras, the network still suffers from an overload situation. Nevertheless, the increased number of neighbouring nodes allows for better cooperation within the network. Therefore, we investigated how the performance can be improved by adding more cameras.

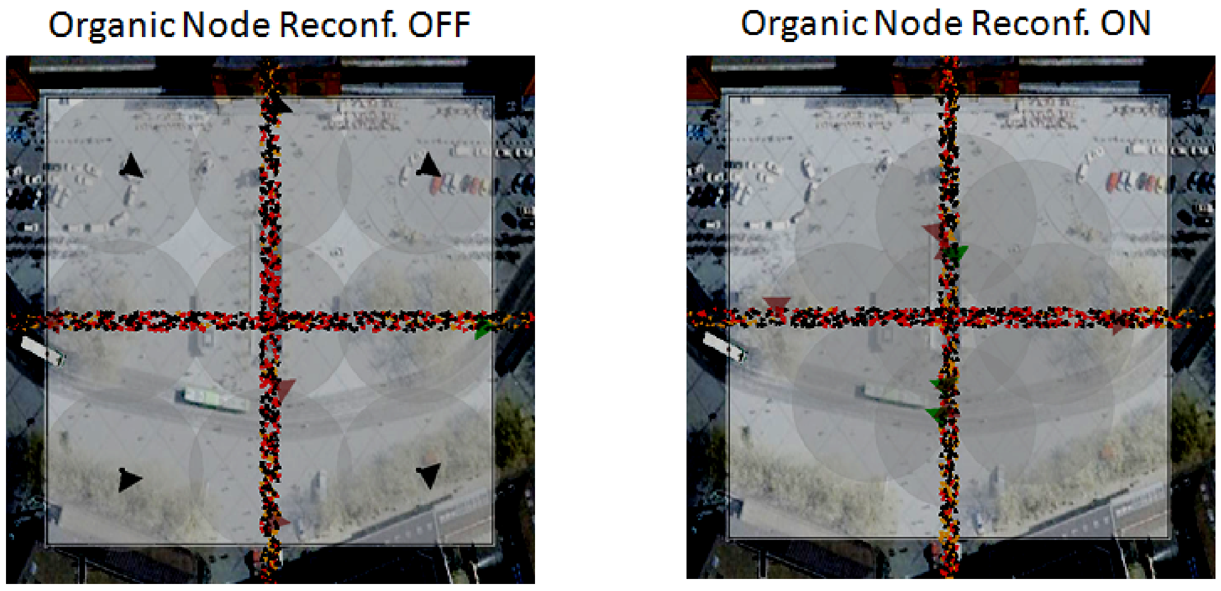

Figure 21: 3x3 Organic Node Configuration

The 4x4 node placement results, shows that with 16 cameras the system can cope better with the overload scenario. Nevertheless, the network still suffers a lack of resources. There is an improvement for spreads up to 50m. Although the difference in TAR values presented in the results of the smaller spreads is not very much.

- TAR behaviour (see Figure 22): With the initial reconfiguration and the node reconfiguration off, the cameras are placed in a way that they can detect targets in the middle small spread trajectories due to the overlap between the nodes. The situation depicted in for small spreads is the same as the explained for 2x2 camera network. Without reconfiguration, the cameras are in their AoM's border observing targets. When reconfiguration is active, more targets appear in their range. Thus, the reaction time increases and the TAR decreases only a 4.6%. For spreads up to 50m, the other cameras that where IDLE cooperate with ones in the center rows. The TAR increases a 21.8% of the original value. Results for spreads up to 150m show an increase of a 3% for TAR values.

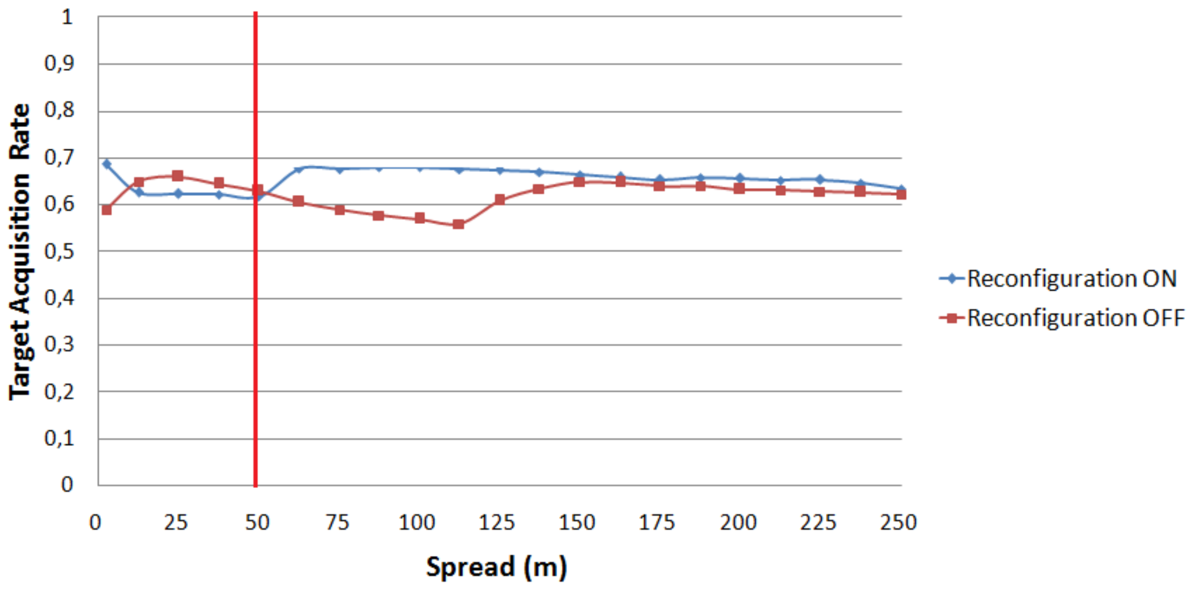

Figure 22: TAR 4x4 Camera Network

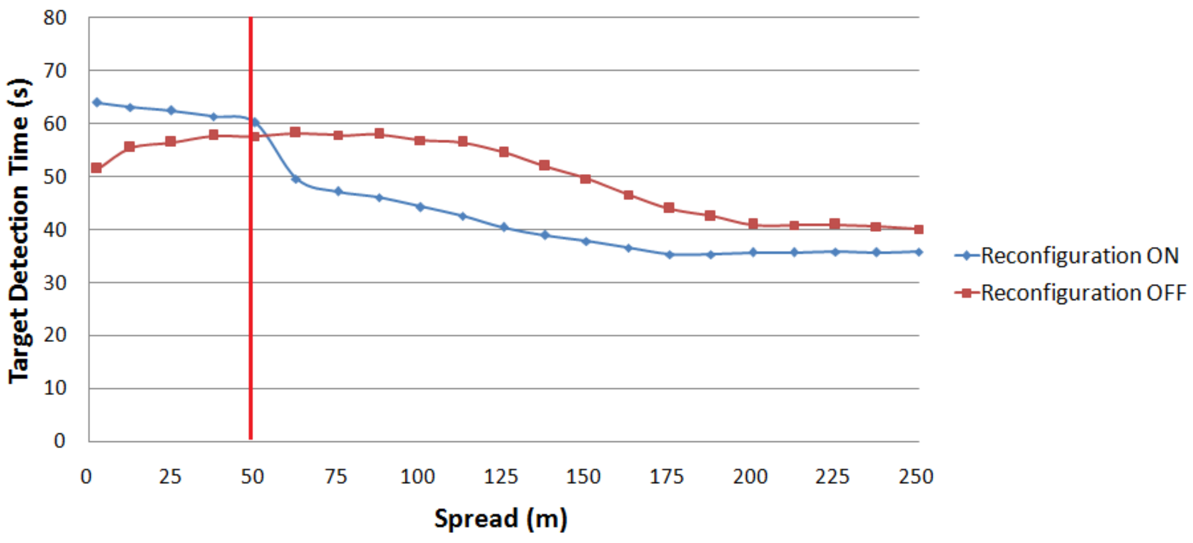

Figure 23: Reaction Time 4x4 Camera Network

- RT behaviour (see Figure 23): The target detection time results present the same situation described above. For spreads under 50m, only the center ACs are observing targets. Due to the situation explained in the 2x2 camera network, the RT is increased a 14% for small targets. Nevertheless, cooperation between all the cameras starts for spreads up to 50m. Thus, the network reduces the reaction time approx. a 26%.

Figure 24 presents the behaviour explained above. For small trajectories in the center of the scenario there is not much difference, between initial placement and reconfiguration. But for trajectories up to 50m width, the cameras that are in the corners, cooperate with the neighbours increasing the overall system's performance.
To increase cooperation, the best solution seems to place the cameras with the center row at the center of the scenario. The 4x4 placement, the four central nodes have 8 neighbours. But some of this neighbouring nodes are the neighbours of each of other center nodes too. Thus, cooperation is not as high as in the 3x3 camera network.

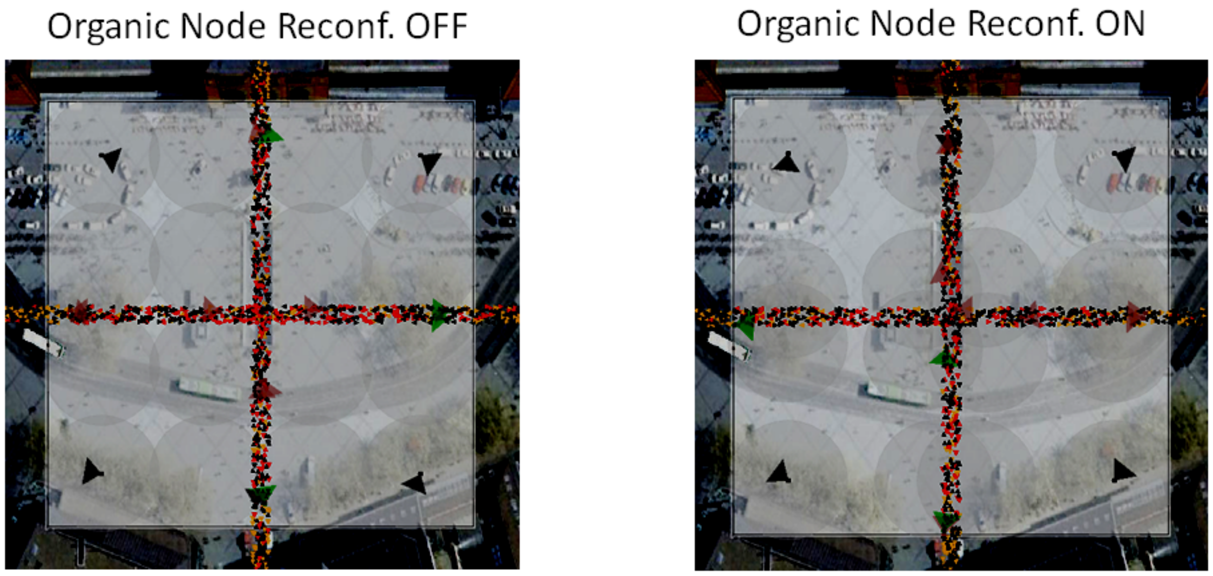

Figure 24: 4x4 Organic Node Configuration

Therefore, we increased the number of cameras to 25. Results presented in Figures 25 and 26 show a great improvement for small spreads. The target acquisition rate and the reaction time are improved significantly in the smallest width trajectories. Finally the system, due to the organic node reconfiguration and the increased number of cameras can handle the overload scenario achieving a TAR between 0.8 and 0.9 as the reaction time for this configuration is reduced a 45% of the original value. Of course as in the other placements, when the spread is increased, the trajectories began to disappear, and the nodes cannot cooperate, so they focus on tracking the targets that surround them, updating their poses and decreasing the reaction time.

- TAR behaviour (see Figure 25): The 5x5 camera network, is similar to the network with 9 cameras. There is a center node with 8 neighbours. Thus, cooperation in the center rows is higher than for the 4x4 camera network. The first area depicted in the figure, shows a TAR increase of 157% of the original value.

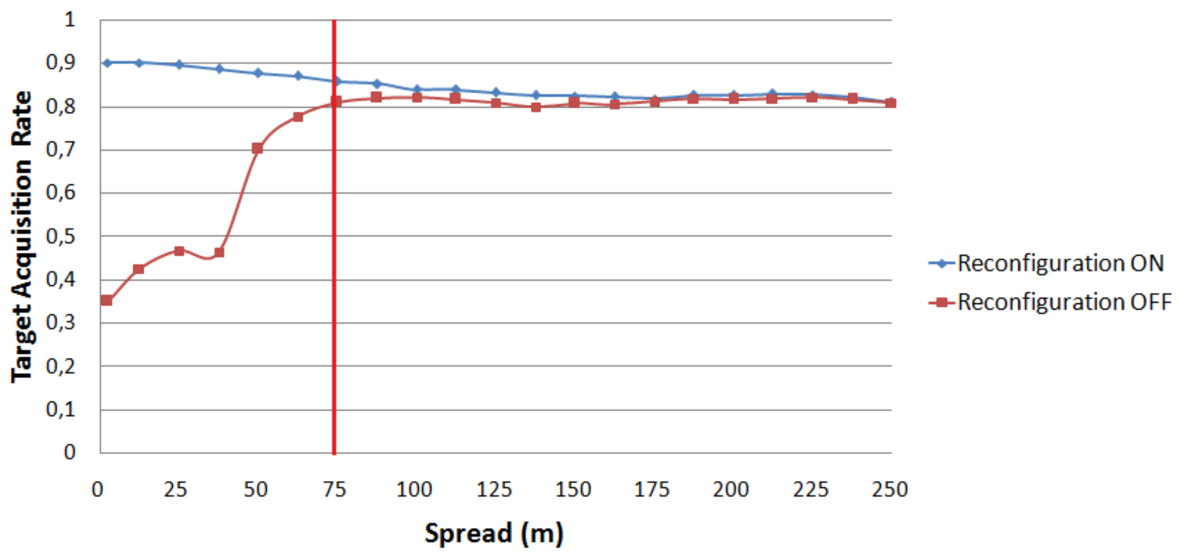

Figure 25: TAR 5x5 Camera Network

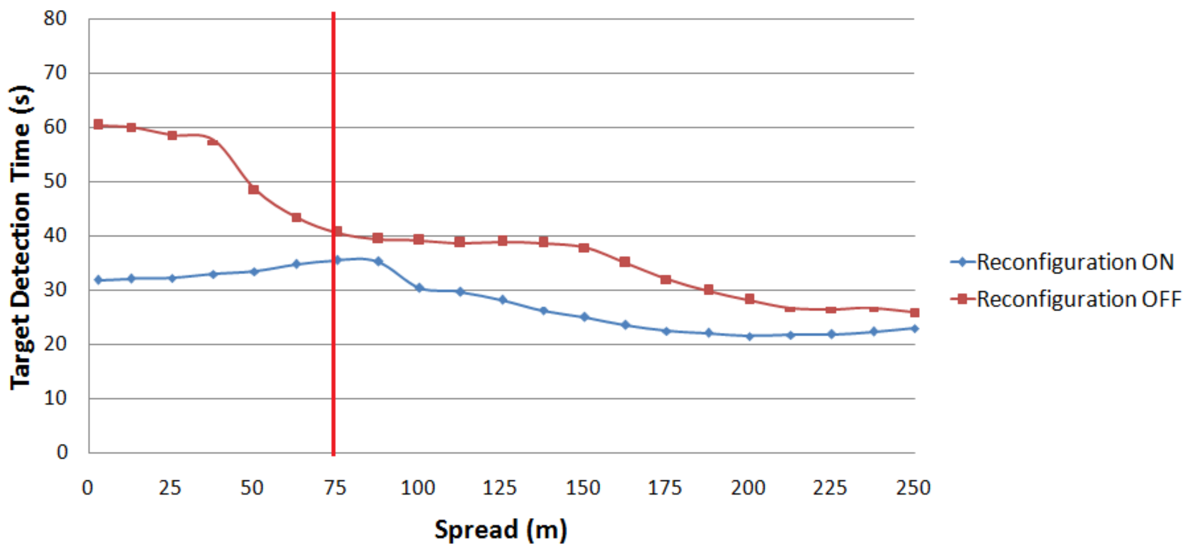

Figure 26: Reaction Time 5x5 Camera Network

- RT behaviour (see Figure 26): Due to cooperation, the reaction time fot small spreads decreases a 45%. For spreads up to 75m where the cooperation is minimal, the RT's decrease is between a 19% and a 34% of the original values.

As in the 3x3 node placement, the nodes are initially placed in the middle of the scenario. Thus, for each node that can detect a trajectory, the number of neighbours is maximal. The Figure 27 shows how, for small trajectories, almost all the cameras are able to cooperate, just 4 of the 25 cameras remain in IDLE mode.

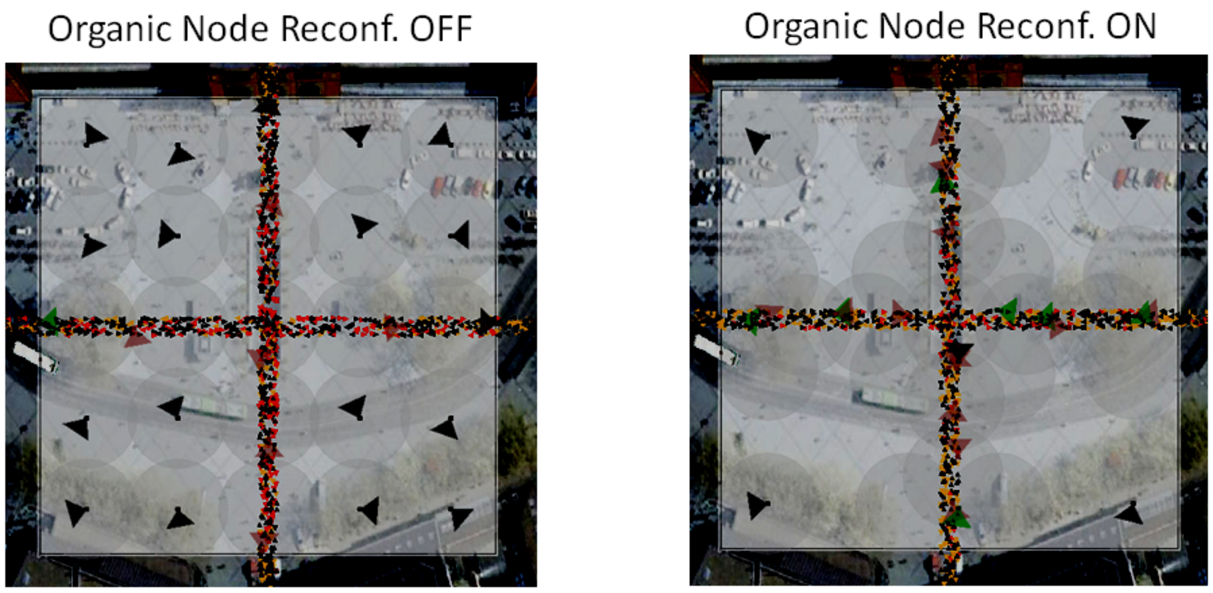

Figure 27: 5x5 Organic Node Configuration

The actuation range radius of the cameras is approx. 25m. To reduce the AoM affects the system in two ways;

- Accuracy: as the AoM has been reduced, there are more cameras covering the whole scenario. Thus, more reconfigurations can be made in a scenario crowded with different trajectories with small spreads. One sector of cameras can focus on a path while in the other corner of the scenario, other cameras are doing the same with another trajectory that goes through this zone.

- Resources: reducing the AoM's range, implies that the maximal movement of the nodes is also decreased. Thus, if in a scenario there is only one thin trajectory in a corner for example, just the cameras near this area are going to cooperate, while the other stay IDLE waiting for targets in their zones.

Although the 5x5 camera network TAR is always over 0.8, we increased the number of cameras to 36 to investigate the system's performance with more cameras but without center row. Results show there is an improvement in TAR also for a 6x6 placement. For trajectories with spread smaller than five meters, the system without reconfiguration shows a TAR under 0.75. Thus, without reconfiguration and with this 6x6 cam placement, the system is not achieving the goal of TAR over 0.8.

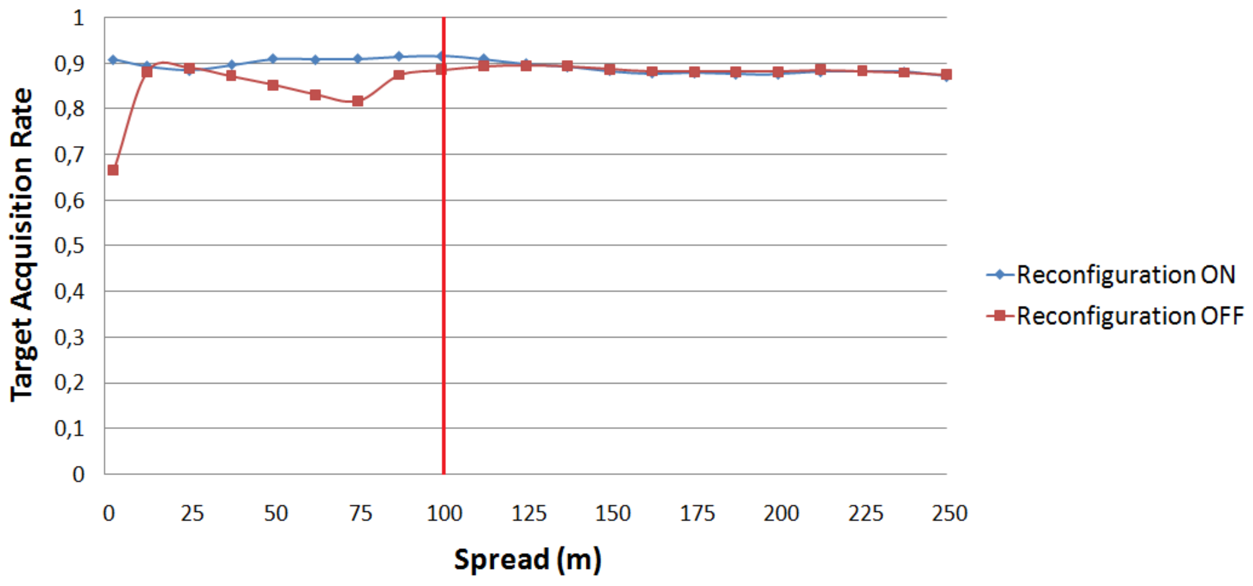

Figure 28: TAR 6x6 Camera Network

- TAR behaviour (see Figure 28): Due to the initial configuration; for small spreads the cameras can detect the trajectories in the overlapping area without reconfiguration as occurred in the 2x2 and 4x4 camera networks. With the organic node reconfiguration, the system reacts more constant to spread changes as cooperation can cope better with middle spread (see trajectories between 25 and 100 meters).
- RT behaviour (see Figure 29): The first situation depicted in results is the same that for 2x2 and 4x4 camera networks. This situation is only for spreads smaller than 25m while for the 16 AC was for spreads under 50m. This is due the fact that the AoM was reduced. Thus, the system can better adapt to small trajectories as the system's accuracy increases as explained above.

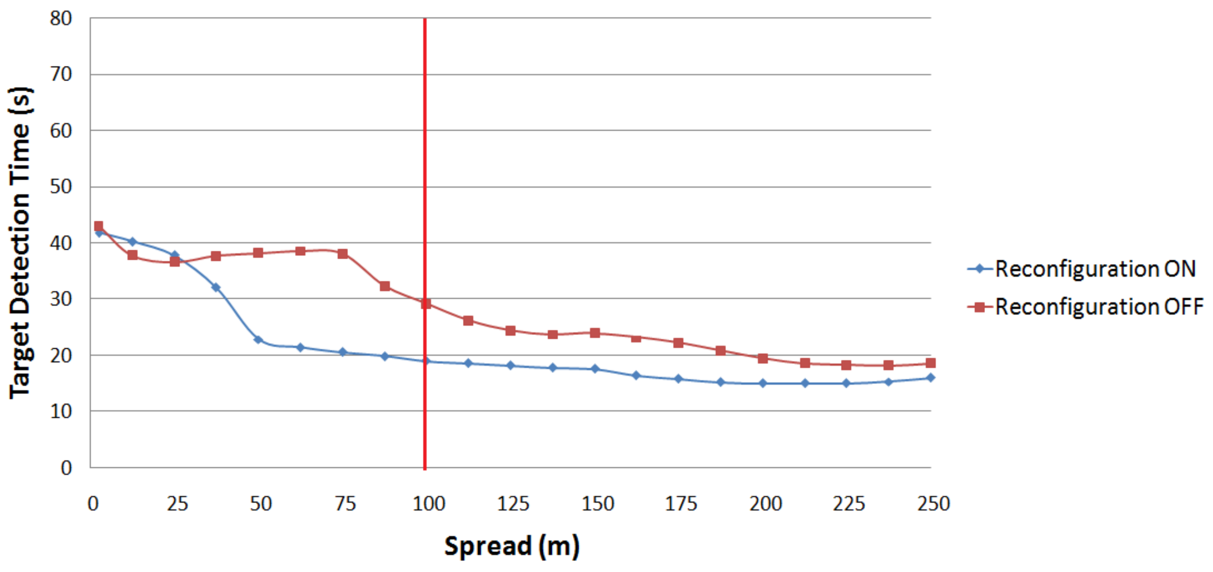

Figure 29: Reaction Time 6x6 Camera Network

This placement presents, how the results can change, if the nodes are not centered in the scenario. For small spreads, the system is using 20 cameras to detect all targets, while in the 5x5 camera network 21 cameras are cooperating to detect the same spreads. As the range of the AoM has been reduced to approx. 20m the nodes can better adapt to scenarios crowded with thin trajectories, but for scenarios with areas without work, a lot of resources can be lost. For the small spreads, the optimal pose of the nodes that can reach the trajectories, is approx. the same that the initial placement. Thus, the improvements, achieved for this trajectories, are not so high as with the 5x5 node placement.

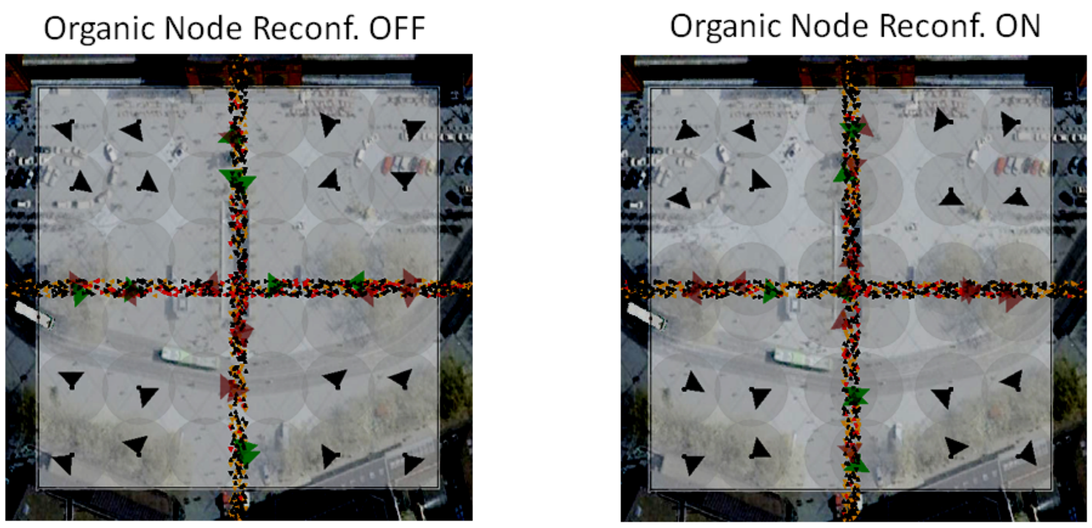

Figure 30: 6x6 Organic Node Configuration

## 4.6 Time-variant Trajectories (with static spread)

This experiment uses the Hannover Main Station as background. There are two different flows of targets going through the surveillance area. The load of these two different trajectories changes over time.

- The initial trajectory emulates the arrival of trains at the station. Thus, targets leave the main station in order to enter the city on two different ways as depicted in Figure 31.
- This situation simulates the arrival of buses at the bus station. The flow of targets changes over time as depicted in Figure 31. After 800 seconds targets do not leave the main station any more but traverse the surveillance area from left to right.

The target generation rate (six persons per second) is the same for both scenarios. The area under surveillance is about 250m x 250m. A 5x5 camera network is positioned on the surveillance area, whereas the target spread is 25m each.

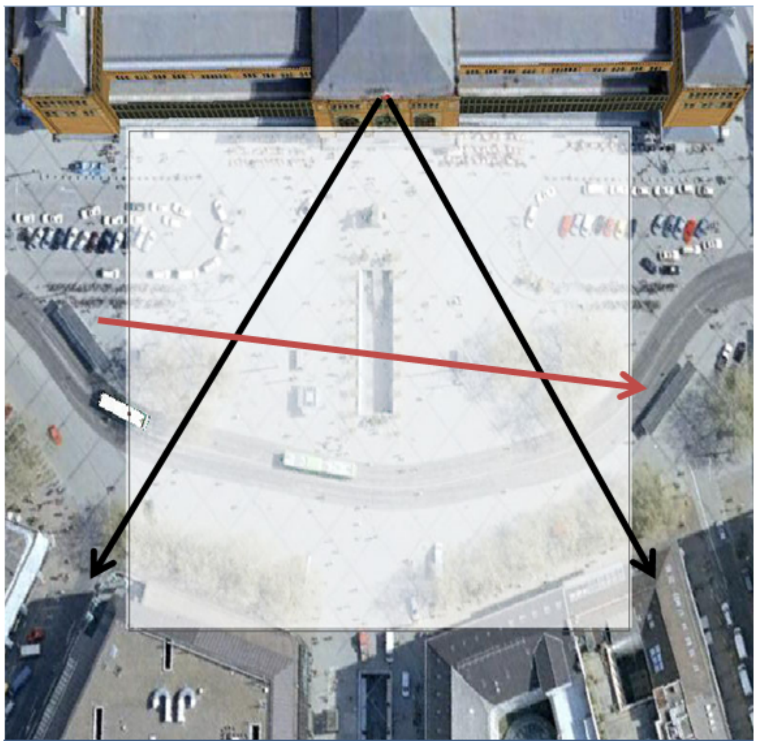

Figure 31: Hannover Main Station

The experiment investigates the following research questions:

1. Which is the optimal Time to Update for the system to reduce the reconfiguration time of the nodes in case of time-variant trajectories, i.e. if the trajectories change over time?

2. How much time does the system need to reconfigure all the nodes to achieve the maximal performance( TAR over 0.91)?

3. In how far is the reconfiguration process influenced by the memory parameter?

To study how all these parameters affect the system's performance a time-variant scenario has been setup. Initially, the first trajectories start traversing the surveillance area. After 15 min, when the nodes have found the optimal position for sure, the load of targets changes to the new trajectory. At this point, we investigate in how far the performance is decreased and how much time is needed to adapt to the new target distribution.

The TTU parameter defines how fast the system adapts to changes in terms of the trajectory distribution.
Nevertheless, in static scenarios, where trajectories do not change over time, the TTU parameter should have a high value, since adaptation is not necessary and the computational cost for re-calculating the best position should be avoided.

On the other hand, in case of highly dynamic scenarios the TTU parameter should be low to adapt the system's resources fast to trajectory changes. Therefore, we investigated how the system's performance depends on the TTU parameter in dynamic environments. The TTU parameter determinates how often the adaptation process is executed.

For small values, the number of samples stored in the nodes memory decreases as the TTU is short. The node computes the optimal position each time and gives depending on the aging procedure higher weight to more recent events. Therefore, for this kind of values, the system adapts better to target distribution changes in the surveillance area.
Our experiment consist of a 5x5 camera network, where each node has a diameter of 50m. Thus, targets traversing a node's actuation range take approx. 33 seconds.
The experiment investigates the target acquisition ratio for different TTU parameters, i.e. 10s,20s,... 50). The starting point of every graph coincides with the point of time where the trajectories change from leaving the main station to entering the surveillance area form the bus station.

Figure 32 presents the results for a TTU parameter of 50s. The red line in the figure describes the TAR ratio that can be achieved for the second configuration, i.e. the trajectories from left to right, in a time-invariant scenario. The node needs 50 minutes (3,000s) to reach this value in case of a trajectory change at time 800. The system needs so much to reconfigure because the data used to compute the optimal pose is based on targets which are not in the cameras' AoM.

Due to the slow TTU parameter, the system cannot adapt faster to trajectory changes. Thus, the target acquisition rate decreases significantly after the initial trajectory is changed and faster values must be investigated.
Since trajectories may change every second in highly dynamic scenarios, a reconfiguration time of 50 minutes is not appropriate. Due to this reason we further decreased the TTU parameter to 40s.

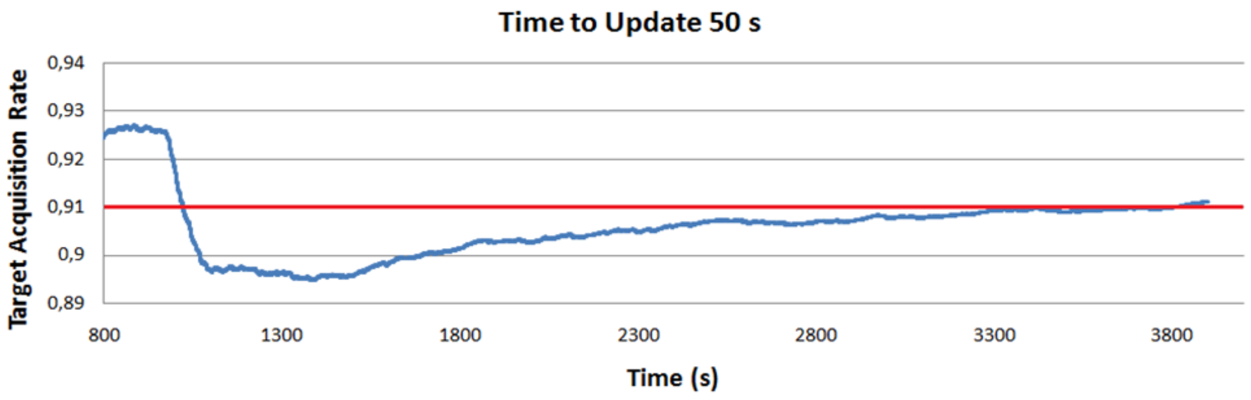

Figure 32: Time to Update 50 s

Due the fact we reduced the TTU parameter, the overall system's performance is improved. The time needed to find the optimal configuration has decreased significantly to 18 min (1,080s) and the target acquisition rates present better results than for a TTU parameter of 50s (see Figure 33). The nodes can compute the optimal pose for their AoM with more recent information. Therefore, the nodes can better adapt to changes and the overall performance increases.

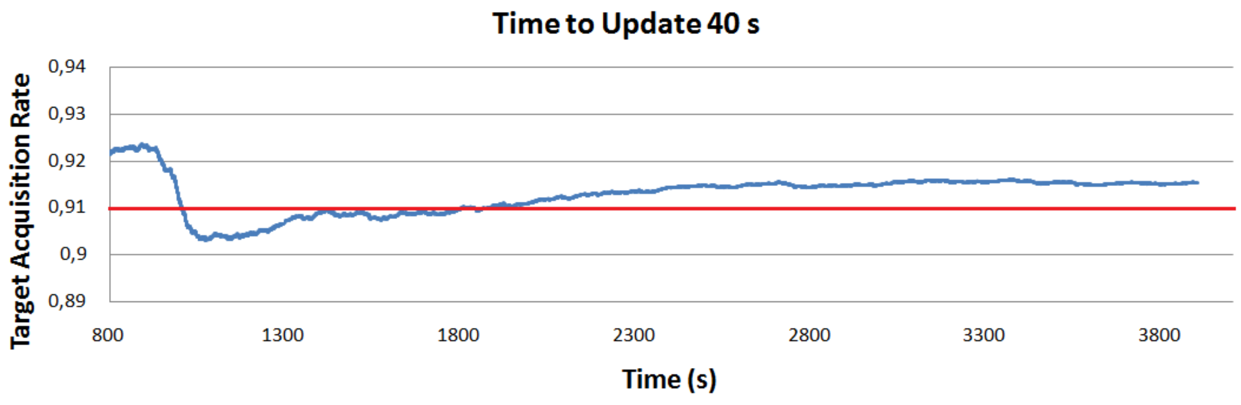

Figure 33: Time to Update 40 s

By further reducing the TTU parameter to 30s the nodes are able to compute the optimal pose before targets are leaving the cameras AoM, i.e. 3 seconds in advance.
Therefore, the system's performance increases, and the time to reach the optimal configuration is reduced to 15min (900). In addition to this, in Figure 34 the TAR value exceeds the performance of the static scenario. This is due, the fast response to changes has been increased. The nodes compute the optimal pose with current targets. Thus, the target detection time is reduced as the current targets are closer to the ACs.

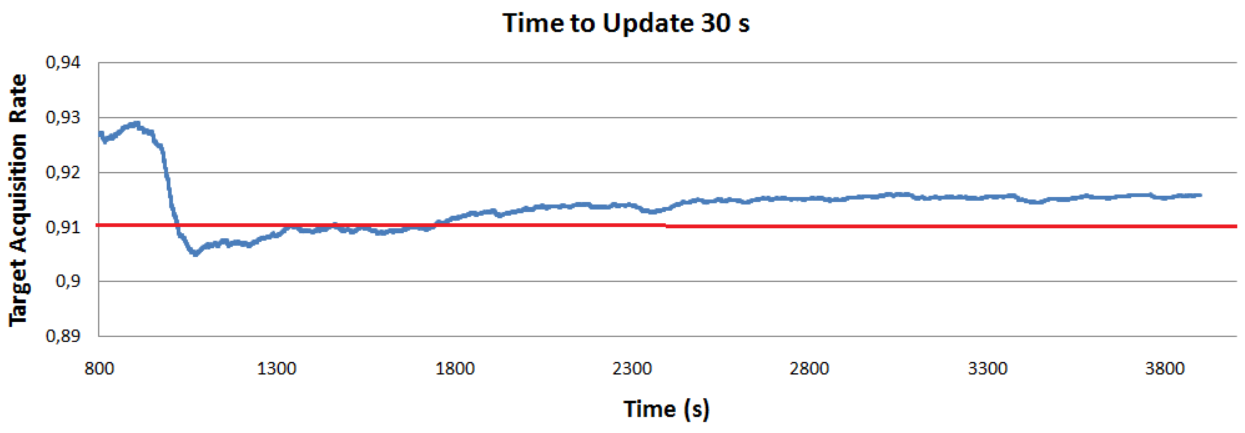

Figure 34: Time to Update 30 s

Due the increased performance for small TTU's values, we investigated how the performance can be improved for values of 20 and 10 seconds. The results are depicted in Figures 35 and 36. Due the fact that the nodes compute their position faster, the system can overcome trajectory's changes faster. In case the load of targets changes to a new trajectory, the nodes reconfigure themselves to detect the new flow of targets. Thus, the system adapts quickly to trajectory changes.
In addition to this, these configurations increase the TAR values even more, in comparison to the static scenario.
For this TTU's values, the nodes update their position two or three times before the current targets leave the camera AoM. Thus, the AoM can change its position to cover higher loads of targets inside the trajectories. Decreasing the target detection time and increasing the overall performance.

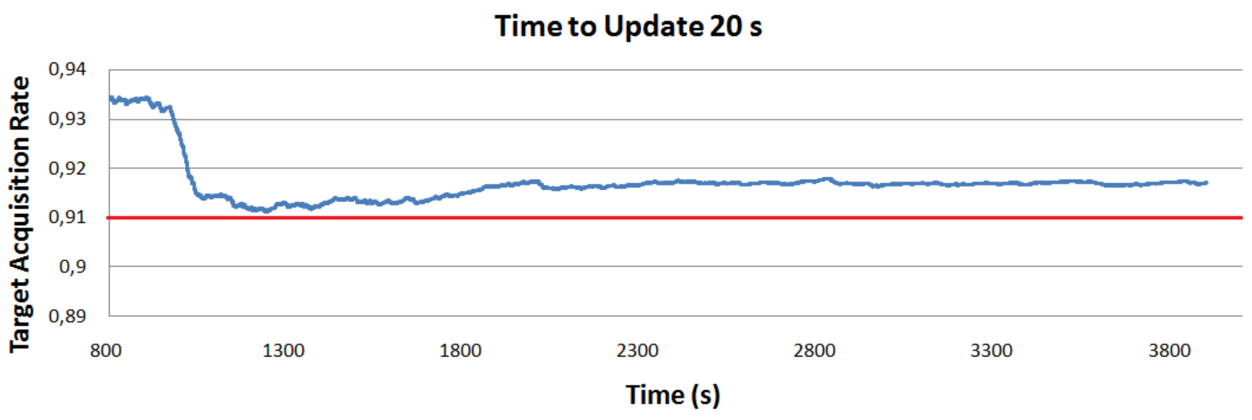

Figure 35: Time to Update 20 s

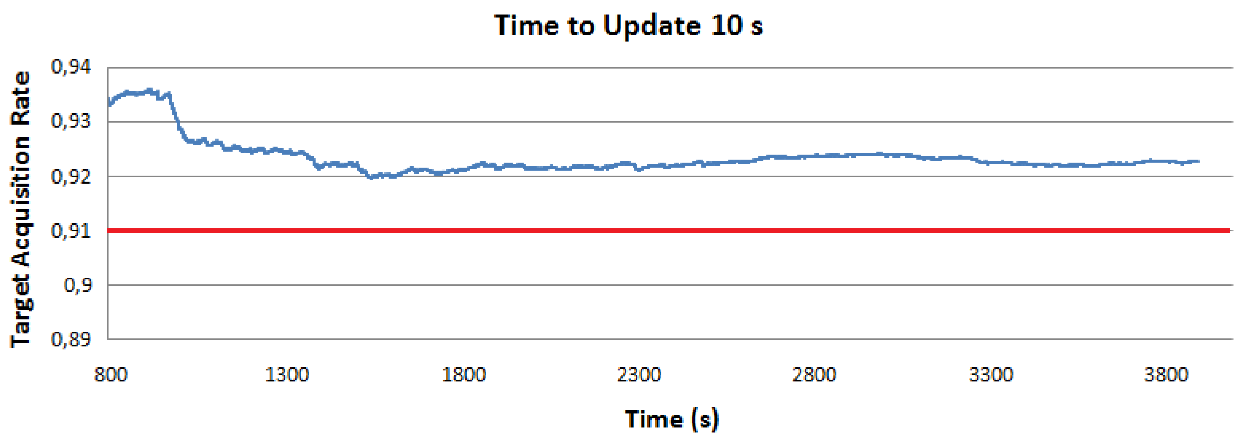

Figure 36: Time to Update 10 s

For smaller TTU parameters (5s, 1s), the target acquisition rate worsens. Due the fact that the number of samples in which the node computes the optimal pose is reduced, and also old samples are affected by the aging factor more often. Thus, the learning process and the accuracy of the node decision worsens and the system's performance decrease as depicted in Figures 37 and 38.

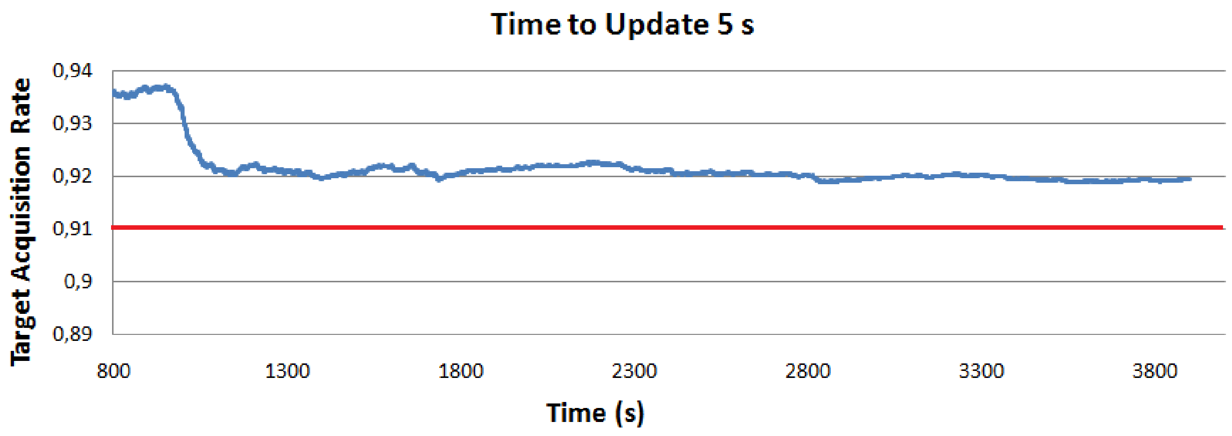

Figure 37: Time to Update 5 s

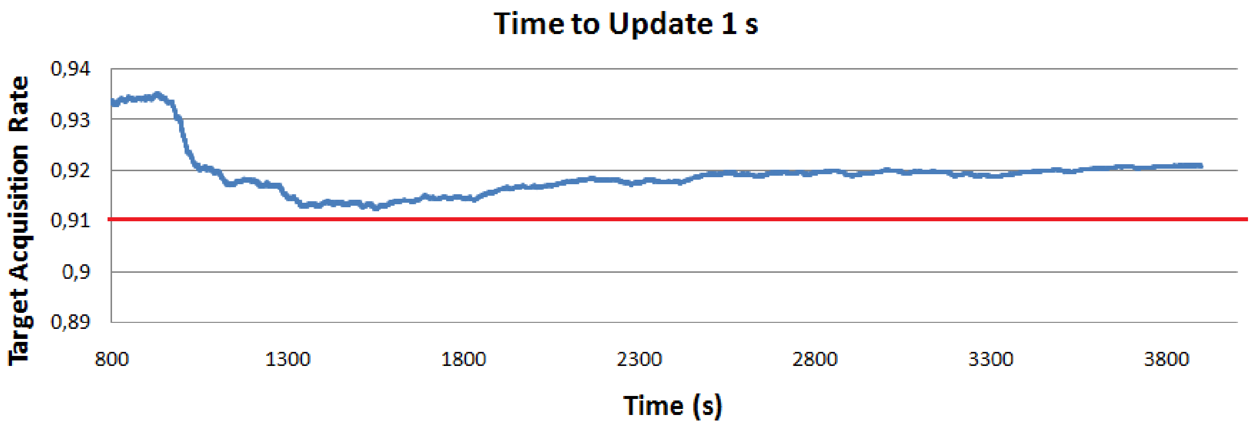

Figure 38: Time to Update 1s

For the small values of the TTU parameter, the nodes compute the optimal AoM's pose with targets that still are in the cameras AoM. Thus, the node can update its AoM in order to observe a higher amount of targets inside of the trajectory. The node adapts its pose to detect flows of targets inside global trajectories.
Therefore, the distance the nodes need to travel to detect targets in their AoM is reduced. This situation can be seen as a local optimal pose optimization.

The nodes adapt to trajectories, and also to the distribution inside the trajectories. Thus, the target detection time is reduced and the overall performance is increased.
Figure 39 shows an example of this behaviour. For a TTU parameter of 50s, the node chooses the mean position from all the targets observed for the last TTU interval. In summary, by using high values for the TTU parameter some kind of smoothing is performed on the target data.

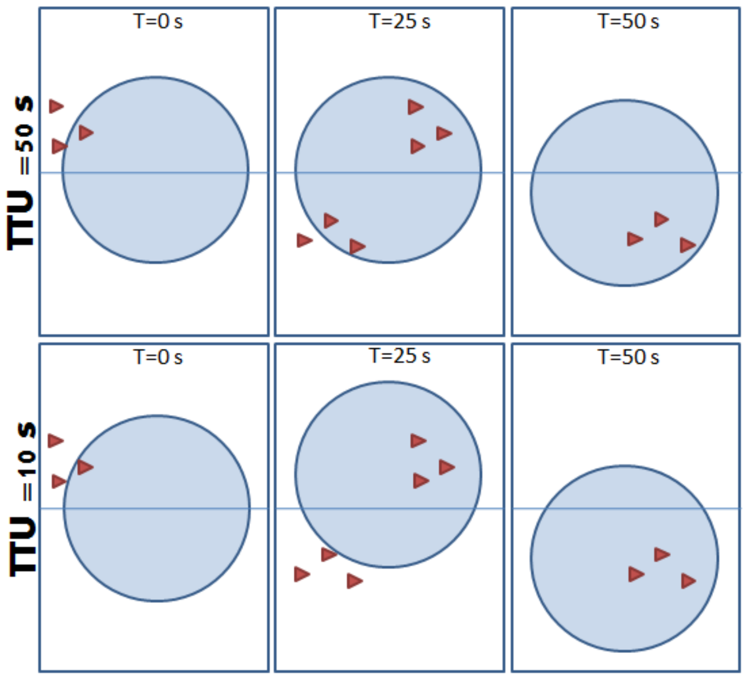

Figure 39: TTU = 10s and TTU =50s

On the other hand, for low TTU values, e.g. 10s the system updates its pose 5 times in 50 seconds. Thus, the smoothing effect is reduced and the nodes react to the noise within the trajectories. Thereby, the overall performance is improved but the reconfiguration cost is higher too.

Once we investigated how the TTU parameter affects to the overall performance, we focused on how the aging procedure (memory) influences to the system's behaviour. We set the TTU parameter to 10s for these experiments to achieve the maximal performance and ignoring the cost.

The memory ratio prevents the utility function from overflowing, since the data is aged periodically as follows. The aging procedure gives a higher weight to more recent events by degrading older data by a factor M each TTU (see Section 3.4.5). Thus, the phenomenon distribution learning continues every TTU. For the following experiment we varied the degradation factor between 0.1 and 0.9.

Figures 40 - 48 present the TAR results for the time-variant trajectories scenario with two different situations; the initial trajectory's reconfiguration and the change of the target load from the initial trajectory to the new one.
For small values of M, the system can adapt faster to changes, but with less accuracy. On the other hand, for high values of M, the system sets a higher priority on past samples. Thus,

the system needs more time for reconfiguration. A detailed explanation of the investigation is presented in the remainder of this section.

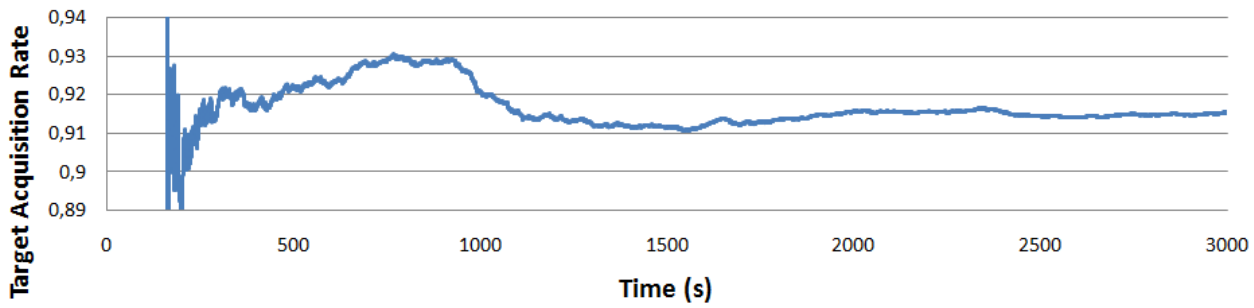

Figure 40: Aging Ratio 0.1

Figure 40 presents the results for an aging ratio of 0.1. The old samples are degradated by a factor of 0.1 each TTU. Thus, the system tries to adapt the position of the AoM with more recent data. After 65 minutes of simulation, the mean TAR values of the last 15 minutes tend to 0,9158.

The results present a good reaction to changes as the old samples do not affect the new decisions. But there is no learning process. Due to this fact, the system can improve its performance if we choose a higher aging ratio.

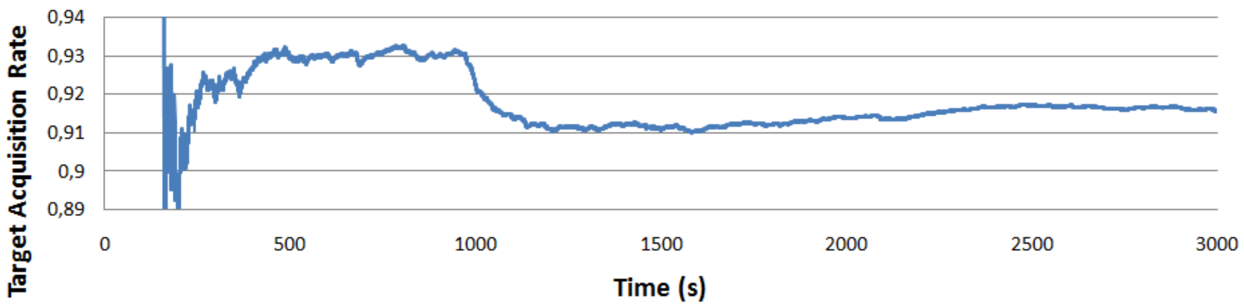

Figure 41: Aging Ratio 0.2

Figure 41 presents the system's performance for an aging ratio of 0.2. The behaviour is similar to an aging factor of 0.1. The TAR values are higher than for 0.1. After 65 minutes of simulation, the mean of the TAR values tend to 0.9161. This is due the learning process is starting to work. On the other hand, the nodes need more data to compute the optimal pose. Thus, we incremented the aging ratio to 0.3.

Figure 42 shows also an improvement in the TAR results. Due to setting a higher priority on older data, the nodes can learn faster where is the new trajectory. Thus, the transition between the change of loads concerning the trajectories is sharper but it needs less time for reconfiguration. In addition to this, the mean of the target acquisition rate for the last 15 min tend to 0.9164 increasing the results for an aging ratio of 0.2.

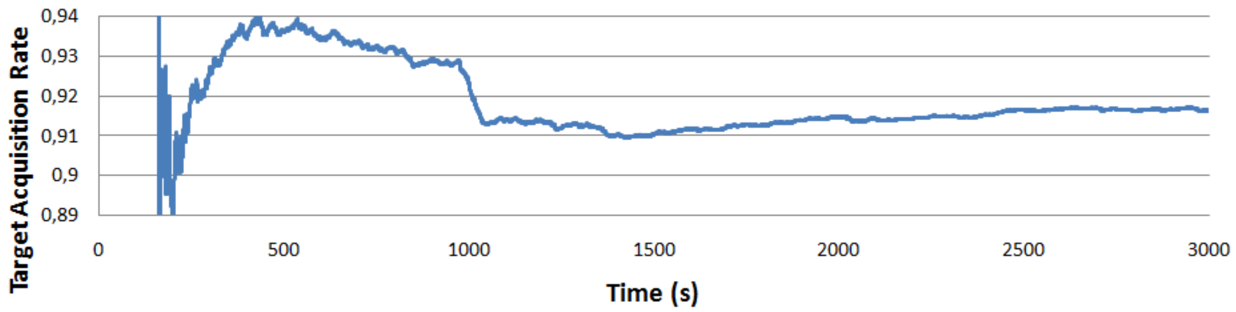

Figure 42: Aging Ratio 0.3

The behaviour for an aging factor of 0.4 (depicted in Figure 43) presents a significant improvement in the overall performance. The mean TAR for the last 15 min of simulation tends to 0.9208. In addition to this, the system overcomes faster trajectory changes. Finally the system increases its performance due to the learning process and the past samples. Thus, we investigated the system's behaviour for an aging factor of 0.5.

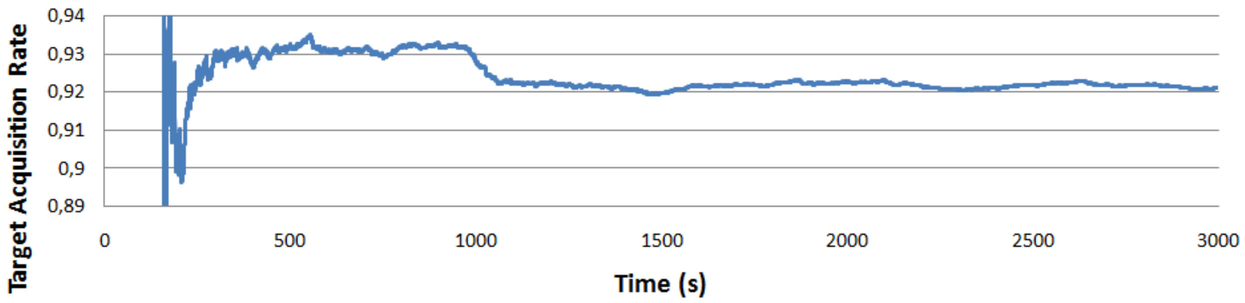

Figure 43: Aging Ratio 0.4

Figure 44 presents the best results for setting the aging ratio to 0.5. The overall performance is increased. As we expected, the tendency for the target acquisition rate increases to 0.923. At this point, the nodes degrade older data by a factor 0.5 each TTU. The aging ratio of 0.5 has the highest effect on increasing the system's performance. The learning process, has enough information from the past events and not too much data to overflow the utility function.

To increase more the aging factor means to start overflowing the utility function. The following results we investigated present, how an overflowed utility function affects the system's performance.

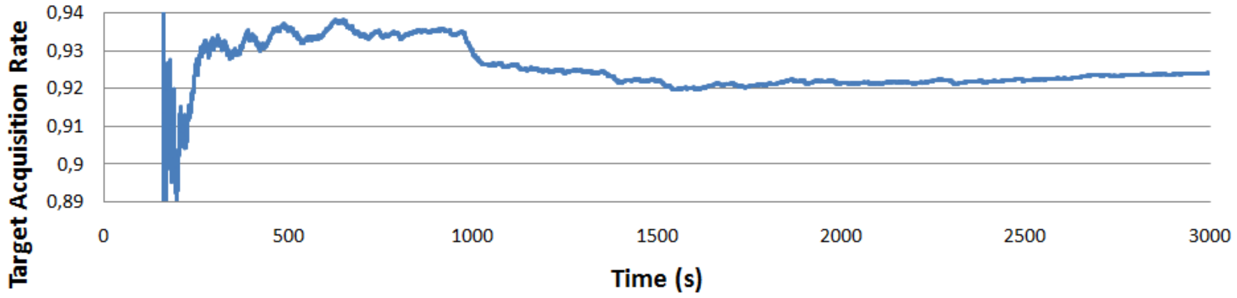

Figure 44: Aging Ratio 0.5

By increasing the aging factor, the system can find invariant trajectories faster. The result for the initial trajectory depicted in Figure 45 presents better results than in Figure 44.
On the other hand, the weight of the old samples in the utility function is higher and this values can overflow the function. Therefore, if changes in trajectories occur, the system's performance decreases (the TAR tendency decreases to 0.9195). In addition to this, the transition between target loads, needs more time for the same reason.

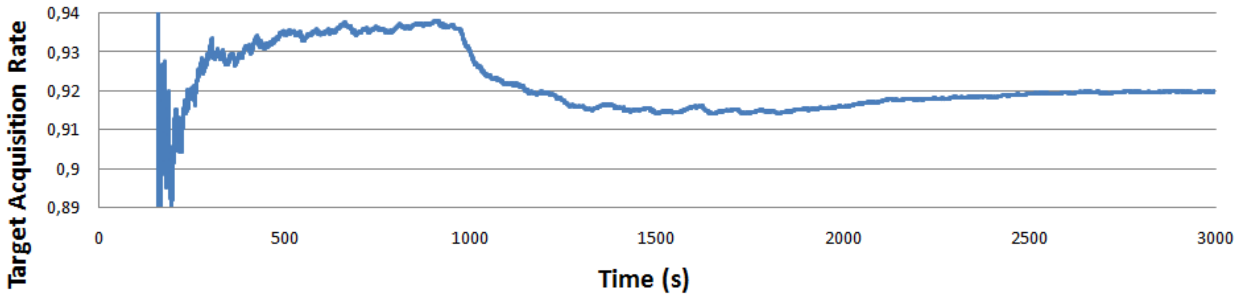

Figure 45: Aging Ratio 0.6

For an aging ratio of 0.7, the system performance decreases considerably. Due the high aging factor, the node computes its decision on new and old samples almost with the same weight. Thus, the target acquisition rate decreases for both trajectories as it is depicted in Figure 46. The TAR tendency for this aging factor decreases to 0.9174.

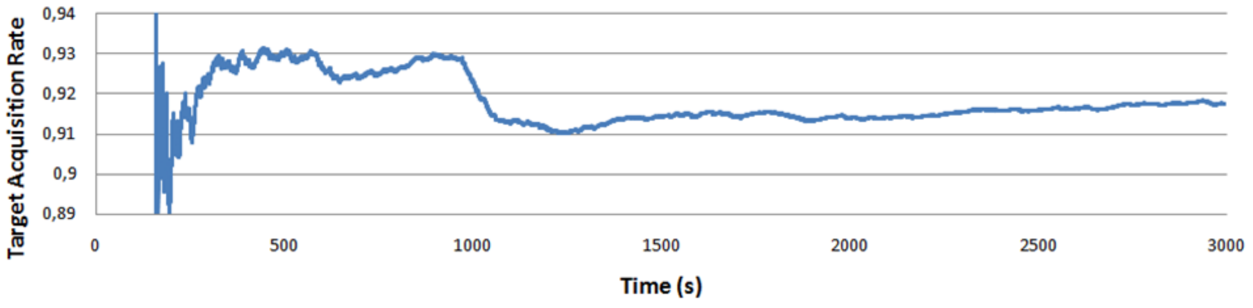

Figure 46: Aging Ratio 0.7

For the last two aging factors (see Figures 47 and 48), the utility function is overflowed. The TAR values decrease almost to 2.3% of the values for an aging factor of 0.5. Each node computes its decision on old and new events. As the second trajectory remains constant, the system can adapt the AoM's position as the number of samples for the new trajectory is higher than the number of targets of the first one. Therefore, as both trajectories are considered with the same weight, the performance would be worse if more trajectory changes occur.

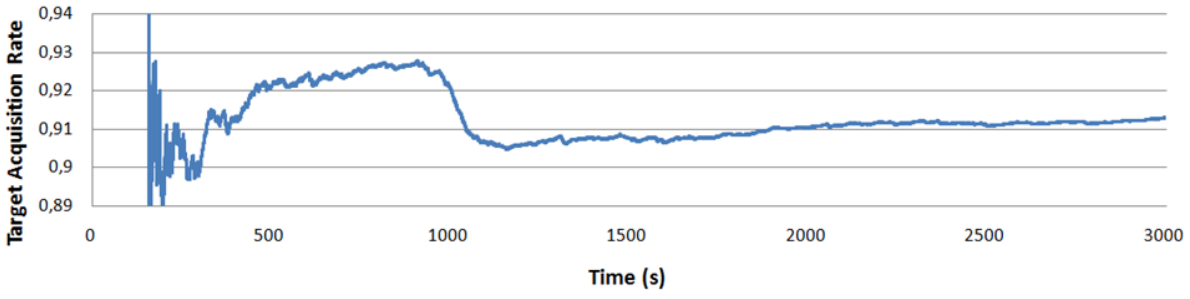

Figure 47: Aging Ratio 0.8

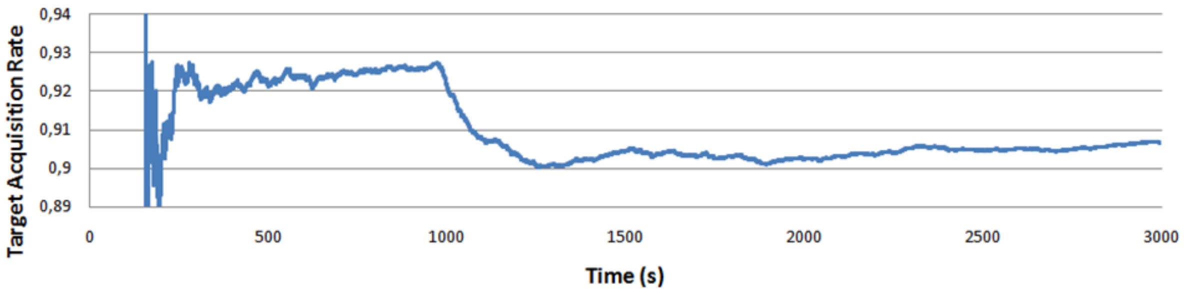

Figure 48: Aging Ratio 0.9

An overall view of how the aging procedure influences to the system's behaviour is depicted in Figure 49. For low values of the aging ratio, the system has no learning process. The nodes compute their decision based just on current targets and not in trajectories.

Therefore, the system can adapt faster to changes but the overall performance decreases.
On the other hand, for high values of the aging ratio, the utility function can be overflowed due to the higher priority on past samples. Thus, the system's performance can decrease significantly if changes in trajectories occur.
As we expected, the best results are for an aging factor of 0.5 where the aging factor does not overflow the utility function and the learning process helps the system to detect trajectories.

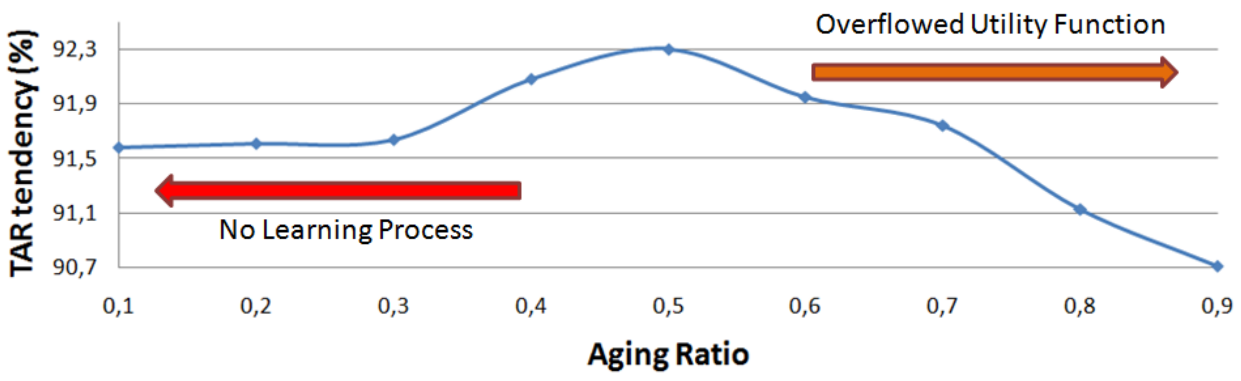

Figure 49: Learning VS Overflow

# 5 Conclusions and Future Work

## 5.1 Conclusions

This section analyzes the results gathered in both simulations. Although there is some fluctuation, the overall trend is as expected. In general, our organic protocol makes way for cooperation between nodes and increases the system's performance.

When a surveillance area is designed, normally most of trajectories will go through the center or near it. The load of targets should be higher in the center than in the borders of the scenario. Thus, the center nodes must have as many neighbours as possible to increase the overall performance.
Due to this, the experiment with the different spreads presents a greater difference between results with node reconfiguration ON and OFF, for trajectory's spreads under 75m and for odd grid placements (e.g. 3x3 or 5x5). For these spreads, the overall performance is improved up to 190% of the results without node reconfiguration.
For widths over 75m, the scenario is overflowed with targets, and the nodes do not need to reconfigure their pose to find the optimal pose. For these values, the improvement achieved by our protocol is small. But this small improvement is driven by the fact that some nodes can find optimal poses where more targets can be detected. In this case, as spread increases, cooperation decreases.

Reducing the AoM affects the system in two ways;

- Accuracy: as the AoM has been reduced, there are more cameras covering the whole scenario. Thus, more reconfigurations can be made in a scenario crowded with different trajectories with small spreads. One sector of cameras can focus on a path while in the other corner of the scenario, other cameras are doing the same with another trajectory that traverse this zone.

- Resources: reducing the AoM range implies that the maximal movement of the nodes is decreased. Thus, for example in a scenario with only one thin trajectory in a corner. Just the cameras near this area are going to increase their spatial redundancy regions, while the other AC stay IDLE waiting for targets in their regions.

The TTU parameter defines how fast the system adapts to changes in terms of the trajectory distribution. Nevertheless, in static scenarios, where trajectories do not change over time, the TTU parameter should have a high value, since adaptation is not necessary and

the computational cost for re-calculating the best position should be avoided.
On the other hand, in case of highly dynamic scenarios the TTU parameter should be low to adapt the system's resources faster to trajectory changes.

The memory ratio prevents the utility function from overflowing and helps the system with the learning process.
This aging procedure gives a higher weight to more recent events than older events by degradation factor M each TTU. The best resutls are for an aging factor of 0.5 where the aging factor does not overflow the utility function and the learning process helps the system to detect trajectories.

## 5.2 Future Work

There are many possible projects that can focus on improvements for the overall performance of the system.

First, to increase the utility of the resources; new nodes with different strategies can be added to the scenario. For example, nodes that could jump between regions or nodes with different W values (i.e. half of the nodes could focus on cooperation and the others on maximal coverage).

Another interesting field of research could be to increase the intelligence of the perceivers and the way targets become salient. Different priorities can be assigned to the targets, e.g. high priority to fast targets and low priority to slow targets. This way, faster targets can be detected in first place while the slow ones are still in the system. Or giving priority to the targets that have spent more time in the system, as may be they are trying to avoid our detection.

A face detection alarm could be implemented. Thus, if someone dangerous has been detected the system gives him more priority and some ACs could be placed in the scenario just for tracking these dangerous targets.

Other possible project could focus on reducing the cost of the system, i.e. optimal camera and sensor networks. Or due to the further evaluation of the parameters for distinct dynamics and scenarios.

# References

[AK06] Stefano Carpin Andreas Kolling. Multirobot cooperation for surveillance of multiple moving targets - a new behavioral approach. In *Proceedings of the 2006 IEEE International Conference on Robotics and Automation*, pages 1311–1316, 2006. 3

[AKS07] Gregory Pottie Mani Srivastava Aman Kansal, William Kaiser and Gaurav Sukhatme. Reconfiguration methods for mobile sensor networks. *ACM Transactions on Sensor Networks*, 3(4):22–28, 2007. 1, 4

[ASW99] Arvin Agah Annie S. Wu, Alan C. Schultz. Evolving control for distributed micro air vehicles. In *Computational Intelligence in Robotics and Automation*, pages 174–179, 1999. 4

[Axt01] Robert Axtell. Social science as computation. In *Technical report, Center for Economic and Social Dynamics*, 2001. 14

[BJ00] Jeffrey K. Bassett and Kenneth A. De Jong. Evolving behaviors for cooperating agents. In *Proceedings from the Twelfth International Symposium on Methodologies for Intelligent Systems*, pages 157–165, 2000. 14

[Bra00] G. Bradski. The opencv library. In *Dr. Dobb's Journal of Software Tools*, 2000.

[CRF01] V. Ramesh C. Regazzoni and G. L. Foresti. Scanning the issue/technology. In *Proceedings of the IEEE (Special Issue on Video Communications, Processing, and Understanding for Third Generation Surveillance Systems)*, volume 89, pages 1355–1367, 2001. 7

[DG07] S. Munder D.M. Gavrila. Multi-cue pedestrian detection and tracking from a moving vehicle. In *Int. J. Comput. Vision 73*, pages 41–59, 2007. 11, 33

[EA96] Joshua M. Epstein and Robert Axtell. Growing artificial societies: Social science from the bottom up. In *MIT Press*, 1996. 14

[Fel] Marco Di Felice. Swarm intelligence: Principles and application in self-organising systems design.

[FP01] F. Fernandez and Lynne Parker. Learning in large cooperative multi-robot domains. In *International Journal of Robotics and Automation*, volume 16, 2001. 14

[GT05] Nigel Gilbert and Klaus G. Troitzsch. Simulation for the social scientist. In *Open University Press, Buckingham, UK*, 2005. 14

[HH07] Martin Hoffmann and Jörg Hähner. Rocas: A robust online algorithm for spatial partitioning in distributed smart camera systems. In *Proceedings of the 21st international conference on Architecture of computing systems*, pages 267–274, 2007. 3

[JS02] B. Jung and G. Sukhatme. Tracking targets using multiple mobile robots: the effect of environment occlusion. In *Autonomous robots*, volume 13, pages 191–205, 2002. 4

[KHLMHA04] Wee Kheng Leow Kian Hsiang Low and Jr. Marcelo H. Ang. Task allocation via self-organizing swarm coalitions in distributed mobile sensor network. In *Proceedings of the American Association of Artificial Intelligence*, 2004. 5

[Mat94] M.J. Mataric. Minimizing complexity in controlling a mobile robot population. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 189–218, 1994.

[MLY04] S. Miller M. Leeser and H. Yu. Smart camera based on reconfigurable hardware enables diverse real-time applications. In *Proc. 12th Annu. IEEE Symp. Field-Program. Custom Comput.*, volume 96, pages 147–155, 2004. 7

[MN03] E. Croft M. Naish. Coordinated dispatching of proximity sensors for the surveillance of maneuvering targets. In *Robotics and Computer Integrated Manufacturing*, volume 19, pages 283–299, 2003. 11, 33

[MWa] A. Duraslan E. Cakar M. Steinberg J. Brehm M. Wittke, U. Jänen. Activity recognition using optical sensors on mobile phones. In *GI Jahrestagung*, pages 2181–2194.

[MWb] J. Hähner M. Wittke. Distributed vision graph update in mobile vision networks. In *23*.

[MWH11] Carsten Grenz Michael Wittke, Sascha Radike and Jörg Hähner. Drofacn: Dynamic reconfiguration of active camera networks. 2011. 19, 28

[O'R87] J. O'Rourke. Art gallery theorems and algorithms. In *Oxford University Press*, 1987. 3

[OW10] L. Wolf O. Wellnitz. Latency in ieee 802.11-based wireless ad-hoc networks. In *Proceedings of the 5th IEEE international conference on Wireless pervasive* *computing, ISWPC'10, IEEE Press, Piscataway, NJ, USA*, pages 261–266, 2010. 33

[Par02a] L. E. Parker. Distributed algorithms for mulit-robot observation of multiple moving targets. In *Autonomous robots*, volume 12, pages 231–255, 2002. 22

[Par02b] Lynne Parker. Distributed algorithms for multi-robot observation of multiple moving targets. In *Autonomous Robots*, volume 12, 2002. 3, 14

[Par03] Lynne E. Parker. The effect of heterogeneity in teams of 100+ mobile robots. In *In MultiRobot Systems: From Swarms to Intelligent Automata*, volume 2, pages 205–215, 2003. 14

[RW08] Bernhard Rinner and Wayne Wolf. An introduction to distributed smart cameras. In *Proceedings of the IEEE*, volume 96, pages 1565–1575, 2008. 6

[SLB] Liviu Panait Keith Sullivan Sean Luke, Claudio Cioffi-Revilla and Gabriel Balan. Mason: A multi-agent simulation environment. 14

[SLP04] Gabriel Catalin Balan Sean Luke, Keith Sullivan and Liviu Panait. Tunably decentralized algorithms for cooperative target observation. In *Technical Report GMU-CSTR-2004-1*, volume 1, 2004. 17

[VTD08] Stefano Nolfi Vito Trianni and Marco Dorigo. Evolution, self-organization and swarm robotics. In *Natural Computing Series*, volume 2, pages 163–191, 2008. 5

[VV05] M. Valera and S. A. Velastin. Intelligent distributed surveillance systems: A review. In *Proc. Inst. Elect. Eng. Vision, Image Signal Process.*, volume 152, pages 192–204, 2005. 7

[Web99] J.G. Webster. The measurement. In *Instrumentation and sensors handbook*, 1999. 11, 33

[WPA] Ieee 802.15 working group for wpan.

# List of Figures