Integration of control, communication, computation, com- plexity and energy considerations in a coherent design strategy

Davide Raimondo, Peter Hokayem, Stephan Huck, John Lygeros, Manfred Morari, Alireza Farhadi, Carlos Canudas de Wit, Sandro Zampieri, Luca Schenato, Angelo Cenedese, et al.

FeedNetback
co-design for networked control systems

GRANT AGREEMENT No.: 223866

Deliverable	Deliverable D6.1
Nature	Integration framework for C4E co-design
Dissemination	Public

T6.1 - Integration of control, communication, computation, complexity and energy considerations in a coherent design strategy

Report Preparation Date	28/02/2011 Project month: 30
Authors	Davide Raimondo, Peter Hokayem, Stephan Huck, John Lygeros, Manfred Morari, ETH Alireza Farhadi, Carlos Canudas de Wit, INRIA Sandro Zampieri, Luca Schenato, Angelo Cenedese, UNIPD Paul Smyth, Jacek Czyz, VICON Giambattista Gennari, VIDEOTEC
Report Version	V2
Doc ID Code	ETH-D06.01-28FEB2011
Contract Start Date	01/SEP/2008
Duration	41 months
Project Coordinator :	Carlos CANUDAS DE WIT, INRIA, France

SEVENTH FRAMEWORK PROGRAMME

Theme 3:

Information and Communication Technologies

Summary

This report is an overview of the research activities regarding WP06 (C4E co-design) of the FeedNetBack project. The objective of WP6 of FeedNetBack is to propose a co-design framework, which allows the integration of control-estimation, communication, computation, complexity, and energy considerations in networked control systems. In this report we outline general guidelines for co-design and illustrate their applicability to the following case studies: (i) surveillance systems using a network of smart cameras and (ii) fleets of Autonomous Underwater Vehicles (AUVs).

Contents

List of Figures

1 Introduction

The objective of Task 6.1 of WP6 is to propose a co-design methodology, which allows the successful and coherent integration of Control, Communication, Computation, Complexity and Energy (C4E) considerations. In WP2-5, we proposed various control strategies take into account communication, computation, complexity and energy considerations individually or pairwise. However, considering these design elements separately may lead to designs that have conflicting goals.

WP6 constitutes a second phase, in which the separate control strategies are integrated into a unified design methodology of C4E co-design. In the process, we analyze the trade-off between the various system requirements, in order to devise a methodical iterative approach that improves the overall performance obtained under all kinds of resource limitations. In particular, we illustrate how this co-design methodology is utilized within the following case studies:

- *Surveillance systems using a network of smart cameras*
- *A fleet of Autonomous Underwater Vehicles (AUVs)*

Such case studies seem to be appropriate in order to demonstrate the wide spectrum of possible applications of the FeedNetBack project: from systems with relatively few, highly mobile nodes, communicating over a network subject to communication imperfections (fleet of underwater vehicles), to systems with a very high number of immobile nodes, with high available bandwidth but also high computation requirements (smart camera network for surveillance applications and motion capture).

To create such a co-design framework we need first to define the specifications of the problem in terms of hardware, software and objectives. Given the specifications, we need to understand the interactions between the C4E design elements, i.e. control, communication, computation, complexity and energy. Then, on the base of specifications and interactions, and given the tools developed in WP2-5, a first co-design is proposed. An evaluation phase quantifies the performance and the trade-offs among the design elements. The C4E design is iterated until objectives are addressed. All of these issues are discussed in the following chapters.

The remainder of this report is organized as follows. In Chapter 2, we propose a novel C4E co-design methodology. In Chapter 3, we review the main issues, design specifications, and interaction among the various design elements for the case studies in WP7 and WP8 of the FeedNetBack project. In Chapter 4, we show how the proposed C4E co-design methodology may be utilized within the scope of the case studies. Finally in Chapter 5 we draw some conclusions on the proposed C4E co-design methodology.

2 C4E Co-Design Methodology for Networked Control Systems

Networked Control Systems (NCS) are becoming ubiquitous, with applications in automotive industry, telerobotics, and deep space exploration, to name a few. Inherent to the design of these systems are the aspects of Control, Communication, Computation, Complexity, and Energy (C4E). While several techniques have been proposed to deal with specific problems related to one or at best two of the C4E design aspects in NCS, it is fair to say that there is yet no coherent methodological framework to address NCS design problems. As such, the main goal of the FeedNetBack project is to devise a novel C4E co-design method, with the aim of improving the overall system performance and quantifying trade-offs between the different design elements.

The C4E design elements cannot be decoupled in general, as seen in Figure 2.1, since one or more of these elements interact and mutually affect each other. As such, any design methodology that tries to decouple these elements is bound to lead to very conservative overall performance of the system and does not fully exploit the potential of NCS. Still, due to the complexity of the overall design problem, most of the available design methods deal with one or at most two of the C4E elements at a time; this is also the approach adopted by WP2-WP5 of the FeedNetBack project. In this report we discuss how multiple design considerations can be integrated, typically in a second design stage following an initial pairwise design.

2.1 Partial solutions to the C4E design problem

Combining the design of two elements in the C4E is a relatively new subject of research. We present three examples of the major advancements in partial co-design involving 2 Cs of the C4E framework, which have appeared over the last decade.

(a) *Control under communication constraints (Control + Communication)* Communication constraints are mainly characterized through three aspects; limited data rates, information losses, and transmission delays. All of these aspects affect the performance of the controller and may lead to instability of the system.

 (i) *Limited data rates:* The designer is given an unstable system and is asked to design coding/decoding schemes with limited bit-rates in order to stabilize the system. Perhaps the most striking result in

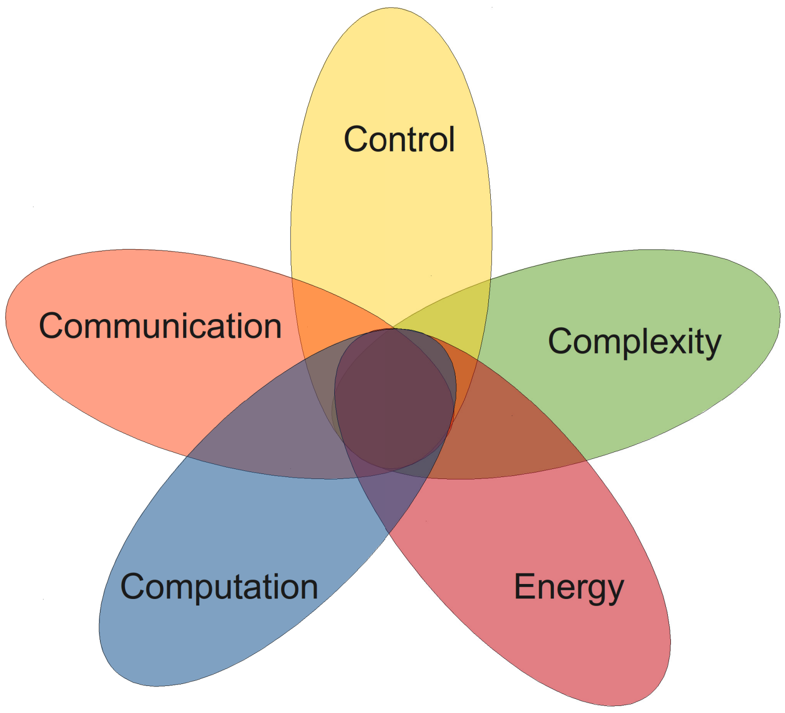

Figure 2.1: C4E framework

this area is the necessary relationship between the required minimal bit-rate and the magnitude of the eigenvalues the (usual) A matrix of a linear system [26]. There are several other results on this subject, which may be found in [16].

(ii) *Information losses:* When the transmission of sensor and control data packets happens over a network with TCP-like protocols, the closed-loop system under LQG controller can be mean-square stabilized provided that the probabilities of successful transmission are above a certain threshold. Since the TCP-like protocols enable the receiver to obtain an acknowledgment of whether or not the packets were successfully transmitted, the separation principle holds and the optimal LQG controller is linear in the estimated state [19].

(iii) *Transmission delays:* This effect is inevitable in scenarios where the controller is not collocated with the plant, for example in telesurgery, or when there are multiple subsystems communicating to achieve a common goal, for example in underwater vehicles. There is a plethora of literature on the subject, in which stability is characterized in the frequency domain for LTI systems, or more generally through Lyapunov-like methods for nonlinear systems [17].

(b) *Control of complex multiagent systems (Control + Complexity)*
Given a large number of systems with 'simple' dynamics, one question is how to design control laws with local information so that a global group behavior emerges, for example, consensus, swarming, formation flight, etc. Here the agents possess limited computational power and limited energy source. Averaging-type control laws emerged as candidates of solving the problem with provable guarantees of consensus, see, for example, [11, 27].

(c) *Computing offline solutions to optimal control problems (Control + Computation)*
Model predictive control (MPC) emerged as a sub-optimal solution to the infinite-horizon optimal control problem with input and state constraints. However, often times the underlying optimization problem to be solved online is computationally demanding, especially when performed with low speed processors. An alternative was to look at parametric solutions that dissect the state space into polytopic regions and a corresponding static affine control law for each region. Both the regions and controllers may be computed offline and stored in a lookup table, and all that the controller is required to do is to measure the state at each time instant, determine the region of the state space in which it lies, and apply the corresponding controller [3].

2.2 Novel C4E co-design methodology

While one is surely to benefit from the proposed sub-solutions above, the question still remaining is how to handle all five elements at the design level. Inspired by [13, 6], we propose an *application-based C4E co-design methodology* that goes beyond the partial solutions listed in the previous section.

Rather than just focusing on one or two elements of the C4E framework, we propose to view the 5 elements in a single framework and apply an iterative co-design scheme. This scheme takes into account the specifications of the problem: limitations of the available hardware/software, computational power, energy limitations, required performance/objectives, etc. Then, through an interaction graph, we highlight which of the C4E elements are of interest and how they interact. An iterative co-design procedure follows this stages. Each of the five elements involved in the C4E framework may be taken into account at the design level through the so-called *critical design factors*, some of which we list below:

Control: stable regulation, fast tracking of mobile objects, minimizing some performance index

Communication: limited bandwidth or bit rate, delays, packet losses

Computation: relatively slow onboard microprocessors and high computation demand with respect to a given sampling rate

Complexity: number of interacting subsystems

Energy: sources (such as batteries) with limited lifetime

2.2.1 Co-Design phases

Our *application-based C4E co-design methodology* comprises three phases:

Phase I: Specifications

In almost every NCS, the designer is given certain specifications, see Figure 2.2. These comprise hardware (i.e. processor, memory, communication bandwidth, physical constraints on sensors and actuators, etc.) and software specifications (i.e. operating system, available programming languages, communication protocol), and objectives/requirements (i.e. efficiency, performance, robustness, flexibility, etc.). These factors are the starting point of our C4E design methodology and affect Phase II, and III below. In principle, one tries to optimize system performance while setting the given specifications as constraints.

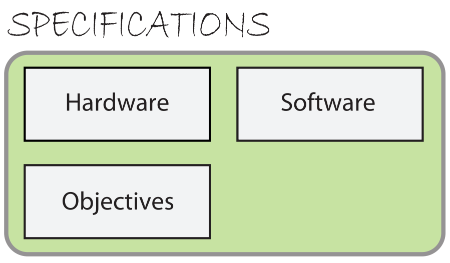

Figure 2.2: Phase I: specifications

Phase II: C4E Interactions Graph

Given the design specifications in Phase I and the nature of the NCS problem, the designer constructs a graph with 5 nodes corresponding

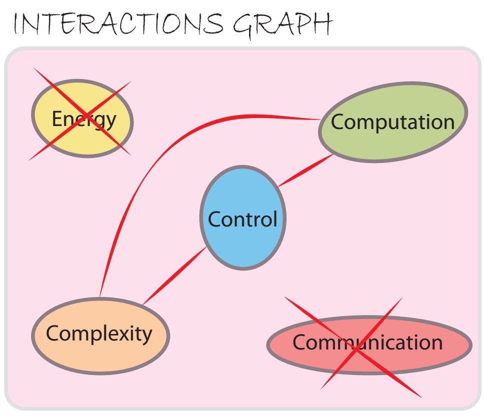

Figure 2.3: Phase II: interactions graph among the various C4E elements

to the 5 elements of the C4E, and the edges reflecting the effect of each of the 5 elements on all the rest. Of course, some of the nodes may not be involved at all and hence they do not have any edges. One generic scenario of this phase is shown in Figure 2.3.

Phase III: Design Iterations

Given the design specifications in Phase I and the nodes and arcs of the interaction graph in Phase II, the designer selects, among the available tools from WP2-5 (WP2 Control & Complexity, WP3 Control & Communication, WP4 Control & Computation, WP5 Control & Energy), the ones that could be of interest.

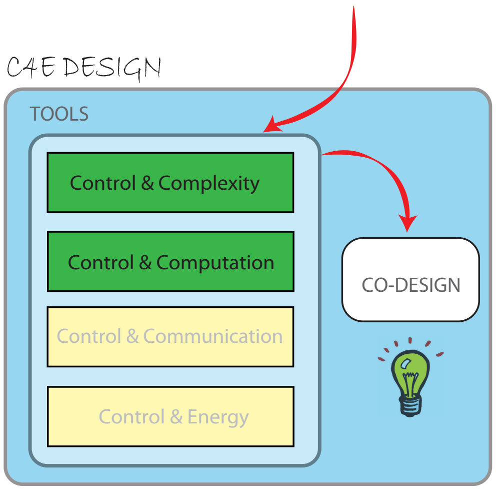

Figure 2.4: Phase III: tools selection and co-design

Hence, a first co-design is proposed, see Figure 2.4. The tuning of selected tools parameters constitutes part of the co-design.

Given the proposed C4E design, an evaluation phase quantifies the performance of the approach and the trade-offs among the design elements, see Figure 2.5. The blue lines represent the proposed strategy. In order to address the minimum required performance, e.g. control quality, a certain complexity is needed, e.g. # of agents. For such complexity, we check if the constraints on communication, computation and energy are satisfied. If all requirements are addressed, then, the design is completed. Otherwise, the evaluation phase provides feedback to the C4E design specifying which design components have to be modified. Modification of hardware and software specifications are also possible.

The overall co-design procedure is summarized in Figure 2.6.

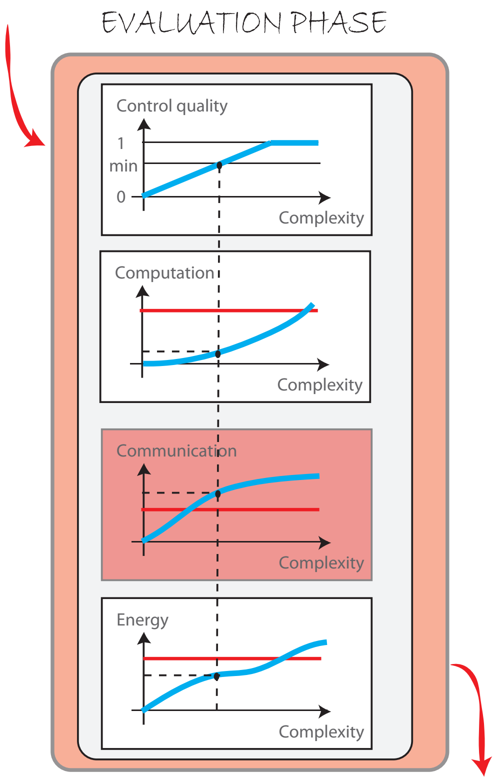

Figure 2.5: Phase III: evaluation

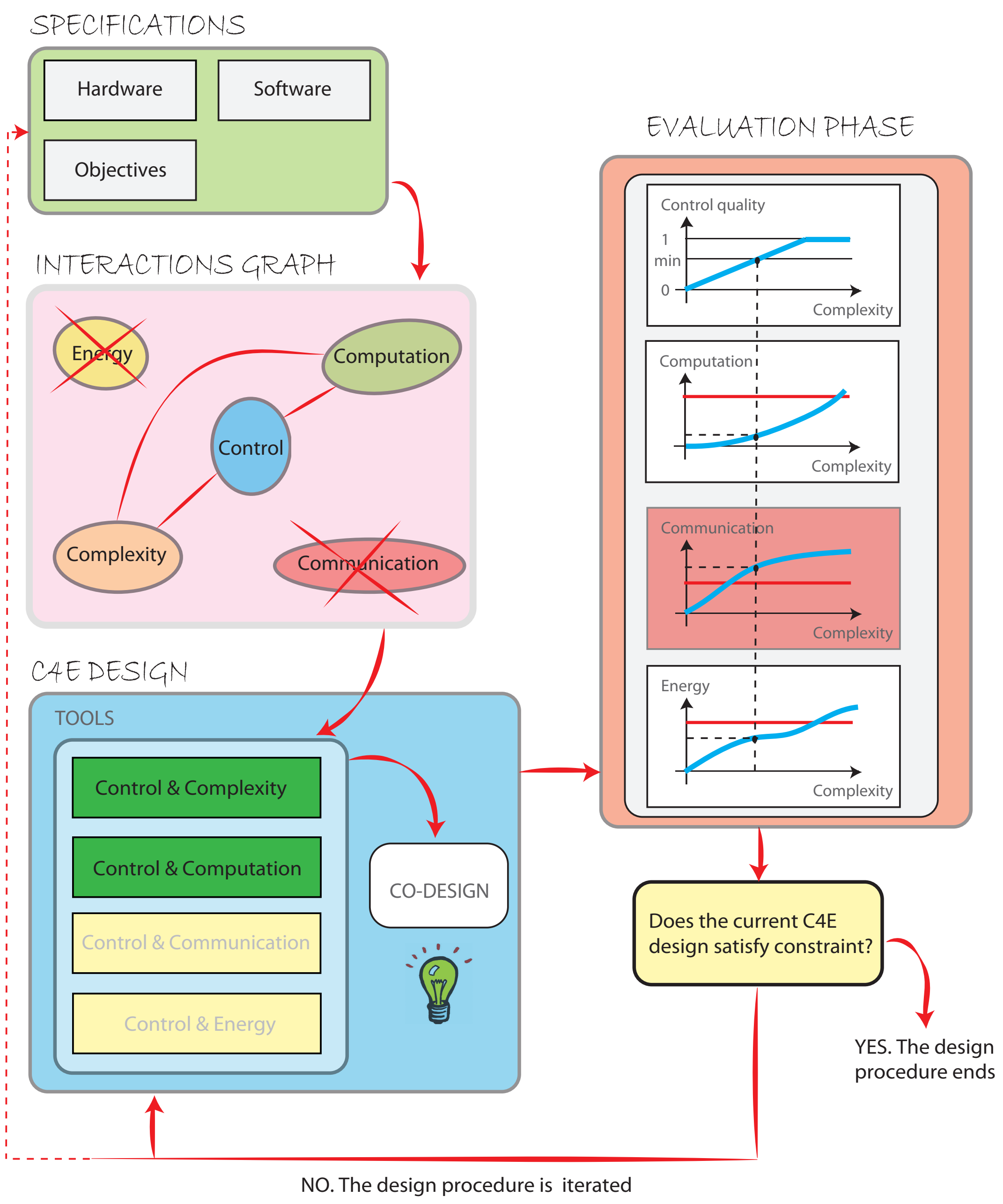

Figure 2.6: C4E co-design methodology: overall scheme. 1) Design specifications are provided. 2) Interactions graph is constructed. 3) An initial design is considered. 4) The design is evaluated on the base of constraints and objectives. 5) The design is re-iterated until we get satisfactory results.

As we shall see in Section 4, this iterative C4E co-design methodology can be applied to the case studies in the FeedNetBack project to improve the performance of the system.

3 C4E Co-Design Issues in the FeedNetBack Case Studies

We review in this section the various case studies within the context of the FeedNetBack project, with the underlying specifications and constraints on each of the elements in the C4E framework.

3.1 Smart networks of cameras for surveillance applications

In intelligent systems for outdoor video surveillance, many Pan - Tilt - Zoom (PTZ) cameras with on-board video processing capabilities are distributed through a site and connected via a LAN network. The volume to be monitored is generally large. The goal is to cooperatively detect and track objects by cameras, as they move through the site.
Target detection involves scanning the monitored site which is called patrolling. After detecting an object, each camera pans and tilts such that an object detected in the assigned camera is in the vantage position in the camera's image plane and consequently its images is captured. When a PTZ camera detects an object it can also focus on such an object and tracks it at a higher level of zoom to gather finer details.
Each PTZ camera is equipped with a Video Agent (VA), where a VA is a logical model of a network-connected processing unit with an active PTZ camera. Video agents are connected to each other and several personal computers via a 100 Mbps Ethernet cable LAN. Video agents are used to actuate/control PTZ cameras. They can directly communicate to each other without the need to communicate to a central server and can send information data and compressed videos to other video agents and one or more remote personal computers in real time, whenever is required.
In general, there are two distinguishable modes in smart networks of cameras for surveillance applications: Calibration mode and cooperative detection/tracking mode, as described below:

Calibration Mode: The calibration process provides the system with the information required for performing detection and tracking tasks. In this process, each camera learns its translation and orientation with respect to a reference (world) coordinate system. Then, each camera learns which are its neighboring cameras. This process also provides knowledge about relationships between fields of view of neighboring cameras as a function of the PTZ states.

For each camera, when a target is detected, its coordinates are expressed in terms of camera pan and tilt parameters and position in the camera frame. Given two cameras A and B, the calibration goal is to estimate a function that maps PTZ states in camera A to PTZ states in camera B when both cameras are watching the same object. In a real scenario, the reference object is in a given location of the monitored site. PTZ cameras are exploring their global field of view to detect the reference object. After a while, all PTZ cameras that can watch the reference object in that location reach a PTZ state such that the object is in the center of the corresponding image plane with a proper size. Consequently, it is possible to establish a correspondence between a PTZ state of a given camera and a PTZ state of other cameras watching the same object. Then, the reference object starts moving and more correspondence are established.

Cooperative Detection/Tracking: Cooperative detection/tracking is a coordinated search of anomalous events. Detection is obtained by moving the cameras properly in order to maximize probability of targets detection at the next time. When cameras perform PTZ movements independently, their field of view may overlap. Intuitively, overlapped field of views must be avoided to cover a wider area at a given time. Moreover, when multiple cameras focus on a same event, wide areas are no longer monitored and other anomalous events will not be detected.
Due to the lack of centralized infrastructure, detection/tracking must be done in a distributed way using local information and communication to achieve global optimal performance. Nevertheless, it is difficult to achieve global optimal coordination starting from local cooperation.
When a moving object is detected, it must be followed by at least one camera. Specifically, the object must remain inside the field of view of at least one camera as much as possible at the maximum zoom.
During tracking process, the 3D position, velocities, and orientation of the target is estimated, and this information is used by the local controller in order to follow the target as it moves inside the whole range of views of the camera under control. Also, when a target is leaving to a neighboring camera field of view, the neighboring camera must rush on the target to keep tracking it. Note that in order to measure the quality of tracking, several performance metrics can be associated with the tracking task such as:
1) The quality of information acquired from a target which can be measured in terms of:
i) Target dimension on the image plane of a camera.
ii) Number of cameras focusing on a target at the same time.
2) The trace reconstruction, which is the number of frames that a target is

effectively and continuously tracked by one or more cameras before being lost.

In the following subsections we study the control architecture of the smart networks of cameras for surveillance applications. We also study the effects of communication, computation, complexity, and energy on such networks. Similar to previous sections, this study is used to understand the fundamental interactions between control, communication, computation, complexity, and energy in the smart networks of cameras for survivance applications.

3.1.1 Control architecture

The control system consists of one local controller, or video agent, per camera. The video agent is a plug - in point to supply control algorithms. Video agents are capable of acting and communicating to each other in a decentralized fashion. In other words, not all video agents have or need the same information, nor the same task. They can autonomously perform a task or cooperate with other agents. In this respect, the surveillance network can be abstracted into a network of multitask and limited resources agents:

- the multitask feature is related to the very nature of the video agent as a smart agent, able to perform different task according to the need and the condition of both the environment and the neighboring agents;
- the resource limitation refers to both the physical availability of functional devices within each single agent (such as mass storage, PTZ unit, and so on), and the non opportunity (if not impossibility) of performing several tasks concurrently.

To this aim, the control architecture regards also a task management issue, since the agents need to coordinate in order to assign and perform the list of tasks that are globally issued to the network. To exemplify the idea, such tasks can be those of tracking and patrolling: the former is related to a specific event that is happening in the scene and has to be focused on, while the latter represents a default state in which the cameras share the task of continuously monitoring the wider possible area in order to detect anomalous situations. The system performance is then obtained through negotiation among agents to track the events, while at the same time attaining a global area coverage [5]. Note that camera PTZ controllers cannot move arbitrary fast. This has to be taken into account when designing control algorithms. The tracking problem has been addressed in the solution proposed in [18].

3.1.2 Communication

Currently, in the smart networks of cameras for surveillance applications, the communication occurs on a dedicated megabit cable Ethernet network which guarantees low latency and packet loss for communication between nearby VAs. However, our industrial partner (Videotec) is interested to use wireless networks for communication in near future. In this type of communication, transmission is subject to noise, short communication range, and limited bandwidth (if VA is equipped with a limited power supply), in which these effects must be considered in the development of control algorithms.

3.1.3 Computation

One of the important issues in the smart networks of cameras for surveillance is the issue of control aware computing. Camera platform is based on FPGA and DSP on-board processors that have slow development cycles and limited computing flexibility. This results in computational latency which needs to be considered when we develop the control algorithms.

3.1.4 Complexity

When the number of cameras or targets increases, the information to be processed and the complexity of the algorithms need to be managed properly. This is done by implementing a decentralized cooperative control technique, in which not all video agents have or need the same information, nor the same task.

3.1.5 Energy

Currently VAs are connected to the power grid through cables. Consequently, the issue of energy consumption is secondary. Nevertheless, in near future, our industrial partner is interested to use wireless communication and solar panels/ batteries as the source of energy. Consequently, it may be necessary to develop communication/computation aware energy techniques.

3.2 Smart networks of cameras for motion capture

One of the case studies in the FeedNetBack project deals with a network of smart cameras for motion capture. Depending on the application, motion capture systems use a few to hundreds of video cameras to capture the motions of physical objects (actors) and translate them into 3D models of the objects. The captured motion is presented in 2D image planes of cameras and translated into 3D trajectories in space in a central PC by a suitable reconstruction algorithms. This is often used to create realistic animations of fictional characters. In these systems reflective markers are placed on the actors' bodies, which are tracked on the 2D image planes of the cameras. In traditional optical motion capture systems a number of cameras surround the capture volume with the requirement that two or more cameras must observe any given marker at all times in order to reconstruct the marker position in space. Each optical unit consists of a distinct video camera, a strobe head unit, a suitable lens, optical filter, and cables. The video camera assembly is a highly advanced unit containing a digital sensor of up to 16 megapixels, on-board FPGA (field programmable gate array) and DSP (digital signal processing unit) and a gigabit Ethernet connection, which are used to perform marker center estimation and 2D marker tracking. The cameras operate on a dedicated gigabit network (a 1000 Base-T Ethernet) for synchronization and data transfer to the central PC. Setup and control signals (of a relatively low bandwidth) are sent from the PC to each camera. The fixed nature of traditional motion capture cameras cannot adjust dynamically to accommodate the motion of the observed objects. The quality of the 3D reconstruction will be improved significantly if the system also uses Pan - Tilt -Zoom (PTZ) cameras (which can rotate horizontally and vertically) by focusing on the objects needing higher details (e.g., hands or face). PTZ cameras also increase the volume of the physical space that actors can move through while being motion captured. In general, there are two distinguishable modes in motion capture systems: calibration mode and operational mode.

Camera Calibration Mode: Camera calibration is the process of determining the approximate pose of all the motion capture cameras as well as their internal optical parameters. The pose of a camera is its translation and orientation with respect to a reference coordinate system which is used by the central PC for reconstruction in 3D. For a motion capture system with fixed cameras, calibration step consists of capturing a trial where the cameras observe a calibration wand with five markers mounted on it in known relative positions.
Calibration for a system consisting of a mixture of fixed cameras and cameras mounted on PTZ heads is very similar to the traditional setup. A subset of the connected cameras will be mounted on pan - tilt heads which can be controlled from the central PC. The standard calibration step will be performed with the PTZ cameras pointing towards the center of the volume and not moving. As a result, the internal parameters and initial pose for all cameras are determined.

In this system, in addition of calibration wand, some markers are placed in unknown positions around the volume. These markers are called beacon markers and will be used for finalizing the calibration of PTZ cameras, as described below:
The PTZ controller has a built in encoder that reports an approximate pose for the camera in a reference frame determined by the PTZ controller software. In order for this pose estimate to be useful we need to produce a mapping of PTZ encoder pose to calibrated camera pose in the reference coordinate system. This pose mapping will be determined automatically after the standard calibration has been performed. The calibration software will exercise the PTZ cameras through their entire motion range with the restriction that they keep a number of the beacon markers in view. This will produce a mapping that can convert any encoder pose into an approximate calibrated pose.

Operational Mode: In this mode, distributed cameras capture the motions of physical actors and send the captured data to the central PC. The central PC then translates this information into 3D models of actors. The control systems developed for this mode will direct the PTZ cameras to focus on areas of interest and thereby improve the effectiveness of camera resources.

In the following subsections, we consider this case study in more details by describing its control architecture and the effects of computation and complexity. Subsequently, we identify the interactions between control, computation, and complexity.

3.2.1 Control architecture

The control system is divided into two parts: A local controller per each PTZ camera and a central controller (running on the central PC) to provide oversight. For controlling purposes, system supplies two communication routes: There is a Local-Central communication path for general use which allows the local controllers to communicate information to the central controller. There is also a Central- Local communication path for communication to the local controllers. The two classes of controllers have different information available and different objectives, as described next.

Central Controller: The central controller has all known information available such as 3D points, and labels (pre-determined identifiers for specific markers on objects), camera poses, 2D to 3D associations and 2D tracks, and 3D world object poses. The objective of the central controller is to direct local controllers to the areas of interests. Central controller also makes decisions as to which PTZ camera to assign to which object, or to multiple objects if multiple objects are visible from a single PTZ camera. The performance of the camera control algorithms is formulated as the minimization of an error metric based on the object pose estimation produced. The object pose estimation algorithm produces an estimate and associated covariance matrix for each visible object. The performance matrix can be then calculated by

$$E = \sum_{i=1}^{m} \min(\mathrm{trace}(C_i), \mathrm{threshold}),$$

where C_i is the covariance matrix for each of the M objects and threshold is the maximum error per object which will be counted if the object is unseen. This error is minimized when all objects are viewed with a good wide baseline by as many cameras as possible. Therefore, the central controller makes a decision for PTZ cameras so that markers are viewed by as many cameras as possible, and from good angles, in order to maximise the confidence in the object pose estimates. Initial steps towards the solution of this problem can be found in [18].

Local Controller: The local controllers are supplied with some static information such as camera internal parameters and mapping of PTZ controller encoder poses to calibrated camera pose frame. For each frame of data observed by the camera, the controller will receive 2D coordinates as well as temporal associations. Additionally, the controller will receive information from the central controller for a frame in the past with a certain latency. This information includes 3D coordinates, 2D to 3D associations and all known camera poses at that time. The local controller's objective would be to follow markers specified by the central controller in the presence of i) Clutter appearing due to background reflections, ii) Processing delays in centrally computing 3D information and camera poses (typically 50 ms).

3.2.2 Communication

As mentioned earlier, for controlling purposes, system supplies two communication routes: There is a Local-Central communication route for general use which allows the local controllers to communicate information to the central controller. There is also a Central-Local communication route for communication to the local controllers. The communication occurs on a dedicated megabit network which guarantees low latency and no packet loss.

The latency of data being captured by the camera until being received by the central PC is less than 5ms. The latency from data capture to processing by the DSP on-board the camera is significantly less. In fact, the temporal and spacial uncertainties such as packet delays, bandwidth limitations, etc. are eliminated by this dedicated network.

3.2.3 Computation

There are processing delays in centrally computing 3D information and camera poses. The amount of the delays D depends on the number of cameras and number of markers/actors involved. Note that, the local controllers will receive information from the central controller for a frame produced almost D ms ago. That is, there is a computational effect that must be considered in the controllers design.

3.2.4 Complexity

Motion capture systems normally involve between 2 and 300 cameras. When the number of cameras increases, the centralized process of reconstructing the 3D positions of markers and the computation of future cameras pose demand a suitable scalability technique (e.g., distributed compution/control techniques) to reduce high computational demands and control complexity.

3.2.5 Energy

The cameras use power-over Ethernet to power cameras and thus the issue of energy consumption for cameras is secondary. Similarly, the central PC is supplied by the city power line and therefore the issue of energy consumption for the central PC is also secondary.

3.3 Fleet of autonomous underwater vehicles

One of the case studies in the FeedNetBack project deals with a coordinated group of Autonomous Underwater Vehicles (AUVs) supervised by Autonomous Surface Vehicles (ASV). The objective is to localize the origin of an underwater source flow located at the bottom of ocean (e.g. fresh water source flow) and consists of two parts, to identify where the source is located and to determine the best way of moving towards it. The localization part is based on detecting the concentration of the source flow using sensed data provided by scientific sensors (payload sensors). The obtained measurement are used to build up the gradient shape and estimate the origin of the source flow. The part for moving the vehicle fleet considers constraints on mission time, fleet formation and communication and is mainly determined by following the gradient and hence approaching the source. Obviously the two parts are coupled since movements of the fleet affect the obtained measurements and, vice versa, the measurement based estimation determines the vehicles trajectories. This mission is called gradient referenced search mission and is a cooperative fleet control mission which has a considerable potential for network control theory. For now the cooperative fleet control law which is responsible for the formations movement generates a (uniformly distributed) circular formation of agents (AUVs) whose center and radius are given by a time-varying reference.

Each AUV is equipped with a powerful on-board computer and uses batteries with limited life time (typically between 5 to 10 hours) as the on-board power supply. Further it is equipped with sensors to measure a level of concentration locally to provide data for the gradient estimation and the generation of the fleets reference trajectory. For the purpose of estimation and formation control, AUVs also exchange their navigation data. The sensed concentration data and navigation data are exchanged via acoustic waves. The corresponding channel is subject to communication imperfections, e.g., short communication range, transmission delay, noise, fading, the Doppler shift effect, etc.

In the following subsections, we study the control architecture of the fleet; and the effects of communication imperfections, computation, complexity, and energy in the fleet control of AUVs. This study is used to understand the fundamental interactions between control, communication, computation, complexity and energy in the fleet of AUVs.

3.3.1 Control architecture

The fleet control law consists of a high level control and a (local) low level control per each AUV. On the one hand the high level control produces the reference trajectory for the vehicle formation so that the fleet eventually reach the source flow. On the other hand, the low level control establishes the desired formation of AUVs specified by the high level control. In case of the (uniformly distributed) circular formation a given center and radius.

3.3.2 Communication

The only practical way for underwater communication is via acoustic waves. Transmission via acoustic waves is subject to significant propagation loss, environmental noise, low speed (which results in propagation delay and the Doppler shift), and multi paths transmission, as described below:

Propagation loss: The propagation loss depends on the range and frequency. The Transmission Loss (TL in dB) increases with range (R in meter) and frequency (f in kHz) as described by:

$$TL = 20\log(R) + \alpha\frac{R}{1000}, \tag{1}$$

where α is the absorption loss which depends on frequency, temperature, and depth.
This loss significantly limits the bandwidth. The practical bandwidth for underwater communication is up to 30 kHz. For a bandwidth up to 30 kHz, the maximum communication range is normally limited to 1km; and for a bandwidth up to 10kHz, the maximum communication range is normally limited to 3km.

Environmental noise: The noise received by the receiver bandwidth limits the maximum acoustic range of the system. There are two kinds of noise: The ambient noise (e.g., hydrodynamics noise causes by wind, rain or biological sea-life noise) and the vehicle noise.

Low speed transmission: The nominal value of sound speed is 1500m/s. The low value of sound speed has two main effects: The propagation delay and the Doppler shift effect distortion, as described below:
Propagation delay: The propagation delay depends on the sound speed value and the range between acoustic transmitter and acoustic receiver. This delay is important for long distances and limits applications with temporal constraints.
The Doppler shift: The Doppler shift distortion corresponds to a frequency shift. It happens when acoustic systems (transmitter/receiver) are moving. It is described by:

$$f_r = \left(\frac{v + v_r}{v + v_s}\right) f_s, \tag{2}$$

where v is the velocity of acoustic wave, v_r is the velocity of receiver relative to the medium (it is positive if the receiver is moving towards the source),

v_s is the velocity of the source relative to the medium (it is positive if the source is moving away from the receiver), f_r is the frequency at the receiver, and f_s is the frequency of the source.

Multipaths: The ocean surface varies from a glossy smooth reflector to a very rough and turbulent surface that scatters sound in a random fashion. The ocean bottom also has a wide variety of compositions, slopes, and toughness, all of which affect sound transmission. The effects of multi paths include constructive and destructive interferences and phase shifting of the signal which results in fading. The effects of multi paths are more present in shallow water and over long distance communication.

For underwater transmission of data, acoustic modems are used. These modems convert digital data into special underwater sound signals produced by ultrasonic devices. These signals pass through underwater acoustic channels; and then are received by a second acoustic modem and converted back into digital data.
In the fleet of AUVs, two types of communication are possible: Between AUVs, and between AUVs and ASV. For communication between AUVs, omnidirectional transducers are used. The allocated bandwidth is 15-28 kHz, the maximum bit rate is 480 bits/s, and the communication range is limited to 1km. The error level for this range is normally 1 bit error over 10 transmitted bits. For communication between AUVs and ASV, omnidirectional looking upward transducers are used; and the allocated bandwidth is 9.5-13 kHz. The maximum bit rate for this transmission is 100 bits/s, the maximum communication range is 3 km, and the error level for this range is normally 1 bit in 10 transmitted bits. Although the available bandwidths for underwater communication are limited, they are enough for a communication without distortion.
Acoustic modems use different kind of data compression, coding, and correction techniques to increase the quality of acoustic communication. They also exchange modem data (e.g., measured noise level); and subsequently source can be adjusted accordingly. Our industrial partner (Ifremer) developed a packet erasure transmission technique with feedback acknowledgement for exchanging data. In this type of communication receiver knows if the received packet contains errors (using powerful error detection/correction techniques); and subsequently it disregards the packets containing errors. Also, transmitter knows (via feedback acknowledgment from receiver) that weather the transmitted packet was disregarded (or not) by receiver. Here, the erasure probability is unknown and varies with time (due to sudden changes in environment, multi path, etc). Typical value for erasure probability is something

between 0.2 and 0.5.
For the low level control to generate a formation, the navigation data must be exchanged between AUVs. This may result in interference. To eliminate interference, we need to use suitable multiplexing scheme. Underwater communication is wideband. Therefore, Orthogonal Frequency Division Multiplexing (OFDM) is very suitable for underwater communication. OFDM is a multi-carrier modulation technique, in which wideband data is transmitted via orthogonal sub-carriers. This method eliminates the interference. In OFDM, the transmitter and receiver must be synchronized (e.g., the receiver must know the frequency used for modulation). But, underwater communication is subject to the Doppler shift effect due to very low speed of underwater waves. Although the Doppler shift effect is small for underwater acoustic waves, it results in asynchronization between transmitter and receiver; and therefore significantly damages the quality of a communication which is based on OFDM. Consequently, we need to compensate the Doppler shift effect by estimating this parameter using test signals. Using the estimated value of the Doppler shift, the effects of this parameter in OFDM are reduced significantly.
Note that from the practical point of view, the available underwater sound modems do not support OFDM techniques. In fact, up to now the only practical way for underwater multiple accessing without collision is via Time Division Multiple Access (TDMA) scheme. But, TDMA scheme avoids simultaneous transmission from multiple nodes. This results in latency which damages the quality of control. This latency grows up with the number of AUVs. Roughly speaking when the number of AUVs is more than ten, this latency is too long that the circular formation of AUVs is impossible to form. The low speed propagation of underwater sound also results in communication delay. The value of this delay depends on the range between acoustic transmitter and acoustic receiver. The value of communication delay can be also measured using test signals. Therefore, in the coordinated fleet of underwater vehicles, communication delay is a known parameter.
In a coordinated group of AUVs, for the safety reasons and also for global positioning, there is communication between AUVs and ASV. Also, from time to time, the collected data (images, etc.) by AUVs are sent to ASV. Therefore, in these applications, the payload data can also be sent from each AUV to the ASV where the reference trajectory can be produced at ASV and then broadcasted to all AUVs.
As shown in [15] to form a uniformly distributed circular formation with a time varying center which eventually reaches to the source flow, the reference center and radius must be updated using the sampled concentration data of all AUVs. But, communication between two AUVs which are far from each

other is subject to high imperfections (absorption, noise, etc) and therefore not possible for this approach. Hence, the payload data are sent to ASV where the reference center and radius is updated there and then broadcasted to all AUVs. Note that when the fleet of AUVs are far from ASV, it might be necessary to position some AUVs between the fleet of AUVs and ASV to maintain communication between the fleet and ASV. These communications are subject to transmission delays. However, the frequency of updating the reference center and radius does not need to be high; and therefore, the effects of these delays are negligible.
Nevertheless, as mentioned earlier, in general there exist reasons for direct communication between the AUVs. For the formation control the AUVs need to exchange their navigation data. If AUVs are used to maintain a communication link or for estimation algorithm which are distributed among the vehicles it is necessary to exchange both, navigation data as well as environmental data such as the sensed concentrations. Consequently, the communications between AUVs are subject to known time delays, in which the effects of these delays must be taken into account when we design estimation algorithms or the low level control law.

Effective communication constraints: Above discussion reveals that in the fleet of AUVs, communication is subject to:
1) *Short communication range:* The communication range between AUVs is limited to 1km and between AUVs and ASV is limited to 3km.
2) *Noise:* Roughly speaking there is 1 bit error in 10 transmitted bits. Using error detection/correction techniques the communication channel can be modeled by a packet erasure channel with unknown and time varying erasure probability.
3) *Delay:* Underwater transmission is subject to communication delay which can be measured.
4) *Communication complexity:* Up to now, the only practical way for underwater multiple accessing without collision is via TDMA, which results in communication complexity. With an increasing number of AUVs, the latency due to TDMA my be too long, so that a fleet formation is impossible to form.

3.3.3 Computation

Each AUV is equipped with a powerful on-board computer. Using approaches like [15], the effects of computation on the low level control performance is negligible since each AUV sends its sampled concentration data to the ASV where the computation of the reference center and radius for the circular formation are done. Afterwards the updated information is broadcast to all AUVs. However, due to long transmission delays in exchanging data between ASV and AUVs, the available time for processing the received information from the AUVs and updating the reference trajectory by the ASV on-board computer is limited. Besides, the ASV on-board computer is responsible for other tasks. Therefore, the high level control law must be developed in the presence of limited computational resource. Distributing the computational tasks by carrying out the gradient estimation on the AUVs will further impose constraints on the computational complexity of the algorithms run on the AUV on-board computer.

3.3.4 Complexity

In the fleet of AUVs we need to employ a proper number of AUVs. AUVs are expensive vehicles (each AUV costs around 1.5 million euros). Therefore, it is desirable to accomplish the gradient search mission by employing as few AUVs as possible. But, employing a small number of AUVs results in relatively poor sampled concentration data which increases the duration of the mission since estimating the gradient shape becomes more difficult. In fact, by using small numbers of AUVs, the search mission may be too long, that is beyond the life time of AUVs.
Small numbers of AUVs also result in a poor control performance. However, a decreasing number of AUVs corresponds to decreasing communication complexity. As discussed earlier, the only practical way for underwater multiple accessing without interference is via TDMA. But, the latency due to TDMA grows up with the number of AUVs. In particular, if the number of AUVs is large, the latency may be too long such that the mentioned circular formation is impossible to form.
Thus, the number of AUVs affects:
1) Cost of mission (small number of AUV = relatively cheap mission).
2) Duration of the search mission (small number of AUVs= long mission).
3) Quality of control (small number of AUVs = poor control performance).
4) Communication complexity (small number of AUVs = low communication complexity).
Therefore, in the fleet of AUVs we need to employ a proper number of AUVs (e.g., between 3 and 10 AUVs) to have a reasonable communication complexity, good quality of control, and a suitable duration for the search mission.

3.3.5 Energy

The energy consumption of on-board computer, sensors, and communication transducers of each AUV are all negligible compared with the energy consumption of propulsion devices which consume most of energy. Therefore, the life time of AUV does not significantly increase by designing communication/computation aware energy techniques. Therefore, in the fleet of AUVs we need to employ enough of vehicles to accomplish the search mission before finishing the available on-board energy resources by the propulsion devices.

4 Applications of the C4+E Co-Design Methodology

Having defined the new C4E co-design methodology and reviewed the case studies within the context of the FeedNetBack project, we illustrate in this chapter how our new co-design methodology can be used to address the design issues in the case studies. In particular, we illustrate how an iterative co-design approach was utilized within the context of autonomous underwater vehicles and multi-camera surveillance systems.

4.1 Co-design for smart networks of cameras for surveillance applications

The objective of this application is to cooperatively detect and track objects by cameras as they move trough the site of interest. Target detection involves patrolling, i.e. scan the monitored site. Once objects are detected, the cameras have to perform tracking, i.e., move in order to keep the objects in view. In the following we consider in details the co-design for tracking.

Tracking

Making use of the "Application-based C4E co-design methodology" described in Section 2, we first identify the specifications of the tracking problem.

4.1.1 Phase I: Specifications

Figure 4.1 illustrates the tracking methodology that we adopted within the FeedNetBack WP7 framework. We assume that there is a processor on each camera. The cameras should perform iteratively the following tasks in parallel:

- At the beginning of an iteration, the camera takes an image, which is subsequently analyzed to detect the targets and compute their positions in world coordinates.
- The measurement information is combined with the measurements of other cameras (the horizontal arrows designate communication).
- Using the current measurements, the position estimates of the targets are updated and the next prediction is computed.

- Given this information, the cameras compute the next positions according to some criteria. This problem is to be solved in a distributed fashion, which again requires communication in order to exchange the individual results.
- As a last step the new inputs are applied and the iteration is restarted which is expressed by the red arrows.

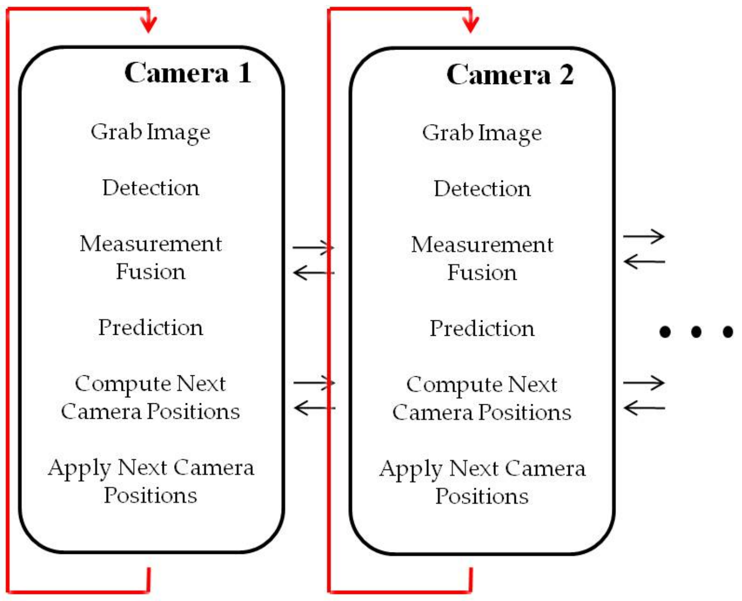
Camera 1
Grab Image
Detection
Measurement Fusion
Prediction
Compute Next Camera Positions
Apply Next Camera Positions
Camera 2
Grab Image
Detection
Measurement Fusion
Prediction
Compute Next Camera Positions
Apply Next Camera Positions

Figure 4.1: Surveillance in a real camera network.

In collaboration with Videotec S.p.A. a test bed comprising pan-tilt-zoom ‘Ulisse’ compact cameras has been set-up. The commands to manipulate the camera position are sent via an USB cable and requested data is received over the same line. This represents a simplified scenario where the communication between cameras is wired and energy consumption is not an issue. The goal of our implementation in the testbed is to emulate the real camera network. It is important to note that the ‘Ulisse’ cameras do not have dedicated processors. The cameras are connected to a central computer that features a ‘Meteor II’ framegrabber card which digitizes the analog image stream. All of the computations are performed on the PC, but in order to emulate the real camera network, we split the program into a thread for each camera. We allow data exchange between the threads as we have assumed communication within the real camera network.

Figure 4.2 shows how the tracking is implemented in our testbed.

- At the beginning of the iteration, the images of both cameras are grabbed.
- Then as explained before, a thread is started for each camera that performs the detection, prediction and position computation.
- After all threads have finished, the position commands are sent to the cameras and the iteration is restarted.

The requirement for a real-time tracking system made it mandatory to use C++ since a Matlab implementation would be way too slow. Besides the desire to emulate a general camera network, the use of a multithreaded approach is furthermore motivated by exploiting the multiple cores of the computer's CPU for computation speed. The testbed comprises two PTZ

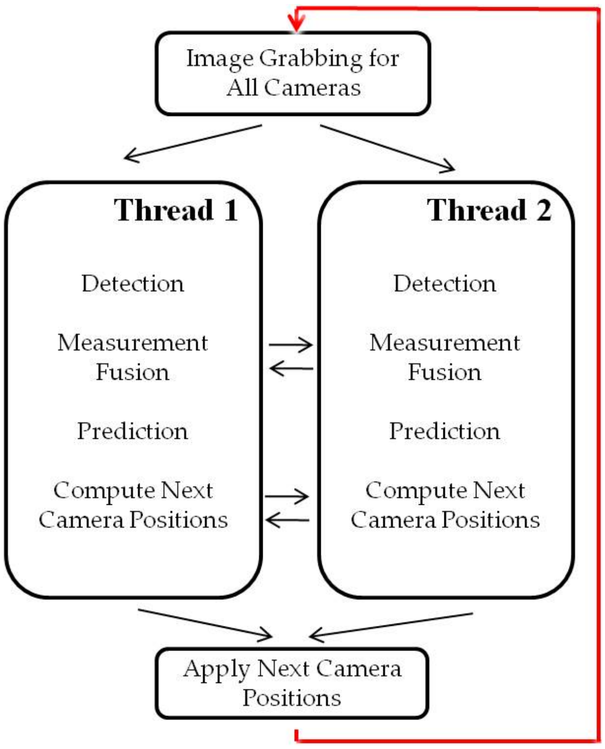
Image Grabbing for All Cameras
Thread 1
Detection
Measurement Fusion
Prediction
Compute Next Camera Positions
Thread 2
Detection
Measurement Fusion
Prediction
Compute Next Camera Positions
Apply Next Camera Positions

Figure 4.2: Tracking in our testbed.

cameras that have constraints on pan-tilt-zoom positions, velocities and accelerations. The relative position between cameras is assumed to be known.

In the experimental setup human driven 1:43 scale remote controlled cars Kyosho dNano and Khepera III robots have been used as targets. The

Kyosho dNano cars can achieve speeds of up to 5m/s, while the maximum speed of the Khepera robots is 0.33m/s. The environment in which the cars move is a white ground plane.

In order to provide a high performance tracking, an accurate model of the tracked objects is fundamental. A unicycle model has been considered for the Kyosho dNano cars and two different discretization method proposed (see the Appendix II): Model(1).A (Euler discretization) and Model(1).B (Trapezoid discretization). After many experiments, we adopted Model(1).B since it described the real behavior of the targets more accurately. For the Khepera robots we used the unicycle Model(2) (see Appendix II).

Performance tracking is also affected by the sampling frequency. A bound on the maximum sampling frequency is imposed by the time needed for computing tasks 1, 2, and 5 of Figure 4.1. The overhead is around 100ms. The time length of tasks *Prediction* and *Computation of the Next Camera Position* depends critically on the particular choice of the controller.

4.1.2 Phase II: C4E Interactions Graph

Previous discussions reveal that in the smart networks of cameras for surveillance applications, there are interactions between control, communication, computation, complexity. If we consider wired communications and grid powered cameras, hence energy and communication will no longer be issues, giving rise to the interaction graph of Figure 4.3. Smart networks of cameras for surveillance are subject to computational constraints and therefore, the effects of these constraints must be taken into account when we design control/computation algorithms. Furthermore, when the number of cameras or targets increases, the complexity of control algorithms must be managed properly (interactions complexity-control). If, on the other hand, we consider wireless communication networks and cameras supplied by solar panels/ batteries, we need to consider the effects of communication imperfections in the development of the control algorithms and we may need to develop communication/computation aware energy techniques, giving rise to the interaction graph of Figure 4.4.

In the rest of this section we will restrict our attention to the case of Figure 4.3 since our test-bed has wired communication and grid powered cameras.

4.1.3 Phase III: Co-Design Iterations

In this section we focus on tasks *Prediction* and *Computation of the Next Camera Position* of Figure 4.1. The choices made to solve these tasks af-

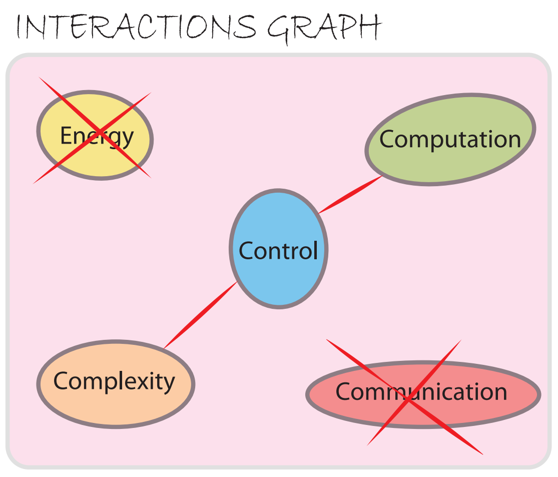

Figure 4.3: Summary of interactions between control, communication, computation, and complexity in the smart networks of cameras for surveillance applications - The case with cable Ethernet

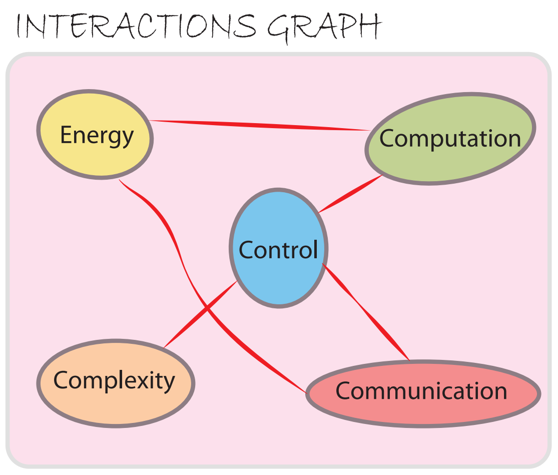

Figure 4.4: Summary of interactions between control, communication, computation, complexity, and energy in the smart networks of cameras for surveillance applications - The case with wireless links and solar panels/batteries

fect Control, Computation and Complexity. According to the time needed for tasks *Grab Image*, *Detect* and *Apply Next Camera Positions* (100ms in total) we choose a sampling time that is adequate in order to track the RC cars or the Khepera robots (control quality). We opted for 250ms. This choice clearly constrains the computational time available for performing the Control step comprised of the *Prediction* and the *Computation of the Next Camera Position* tasks. One has the choice among many possible control techniques for tracking. Depending on the chosen controller, we get different relations between control quality and complexity (i.e., the number of cameras), and computation and complexity (e.g. Figure 4.5). Minimum control quality requirements and maximum available computational power constitute the constraints of the problem.

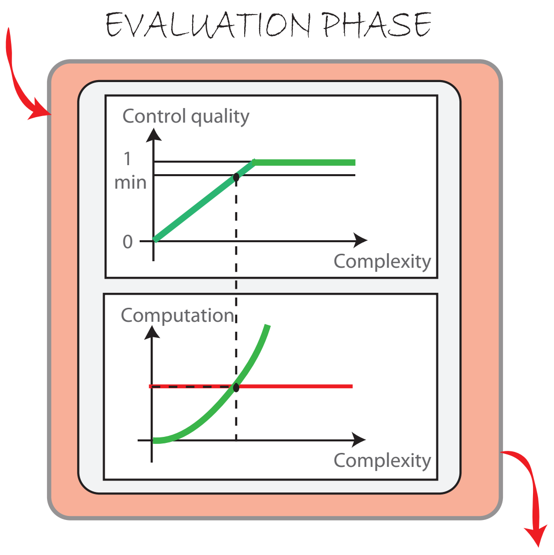

Figure 4.5: Co-design for smart networks of cameras: evaluation phase.

In the following we consider in details the possible tools (see Figure 4.6) available to solve the tasks *Prediction* and *Computation of the Next Camera Position*. Then we discuss how we iterated among the various solutions in order to improve the overall system performance.

Prediction

In the tracking process, the 3D position, orientation, and velocities of the targets have to be estimated. Two filters have been considered: Extended Kalman Filter (EKF) and Particle Filter (see [21] for details).

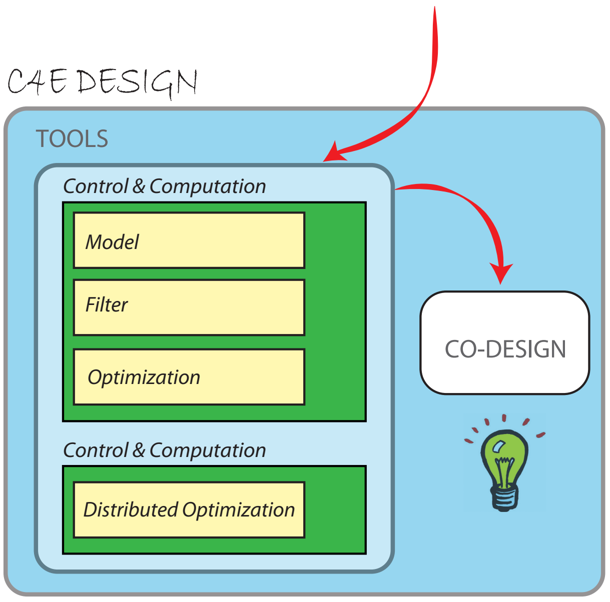

Figure 4.6: Co-design for smart networks of cameras: tools.

Filter.A

Extended Kalman Filter

EKF applies to nonlinear systems

$$\begin{aligned}\mathbf{x}_k &= \mathbf{f}(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}, \mathbf{w}_{k-1})\\ \mathbf{z}_k &= \mathbf{h}(\mathbf{x}_k, \mathbf{v}_k)\end{aligned} \tag{3}$$

by linearizing around the current estimated state. In this system $\mathbf{x}_k$, $\mathbf{u_k}$ and $\mathbf{w_k}$ are the state, input and process noise vectors at time k respectively and $\mathbf{f}(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}, \mathbf{w}_{k-1})$ is the nonlinear state update function. Furthermore $\mathbf{z}_k$ is the measurement taken at time k which is modeled to be the output of the nonlinear measurement function $\mathbf{h}(\mathbf{x}_k, \mathbf{v}_k)$ where $\mathbf{v}_k$ indicates the measurement noise at time k. The process noise $\mathbf{w}_k$ as well as the measurement noise $\mathbf{v}_k$ are assumed to be zero-mean Gaussian distributed

$$\mathbf{w}_k \sim N(0, Q_k) \tag{4}$$
$$\mathbf{v}_k \sim N(0, R_k) \tag{5}$$

where Q_k and R_k are the covariance matrices of the process noise and measurement noise respectively. In the case of model (28), the process noise covariance matrix is modeled as

$$Q_k = Q = \begin{bmatrix} \sigma_{x,y}^2 & 0 & 0 & 0 & 0 \\ 0 & \sigma_{x,y}^2 & 0 & 0 & 0 \\ 0 & 0 & \sigma_\Phi^2 & 0 & 0 \\ 0 & 0 & 0 & \left(\frac{a_{max}}{2}\right)^2 & 0 \\ 0 & 0 & 0 & 0 & \left(\frac{\psi_{max}}{2}\right)^2 \end{bmatrix} \tag{6}$$

where we use the maximum translational acceleration a_{max} and maximum angular acceleration ψ_{max} to calculate the variances of the accelerations. The values of those accelerations can be obtained from the car manufacturer's manual. We want the maximum change to be in the 2σ interval, hence we divide the values by 2 before squaring them. The variances of position and orientation $\sigma_{x,y}^2$ and σ_Φ^2 are tuning parameters of the model.
The measurement covariance matrix is given as

$$R_k = R = \begin{pmatrix} \left(\frac{\Delta z_{x,y,max}}{2}\right)^2 & 0 & 0 \\ 0 & \left(\frac{\Delta z_{x,y,max}}{2}\right)^2 & 0 \\ 0 & 0 & \left(\frac{\Delta z_{\Phi,max}}{2}\right)^2 \end{pmatrix} \tag{7}$$

where the maximum measurement errors $\Delta z_{x,y,max}$ and $\Delta z_{\Phi,max}$ are estimated using repeated measurements of a constant position. Video [28] gives an example of the application of EKF to the surveillance problem.

Filter.B

Particle Filter

Particle filters (or Sequential Monte Carlo methods, see e.g. [9]) are fast estimation techniques that perform a numerical approximation of the pdf of interest using simulation. Consider a generic time $k \geq 0$, and denote $\mathbb{Z}(k) = \{\mathbf{z}(i)\}_{i=0,\cdots,k}$ and $\mathbb{X}(k) = \{\mathbf{x}(i)\}_{i=0,\cdots,k}$ respectively the sequence of measurements and states up to time k. The main idea of particle filters is to approximate the continuous probability distribution of interest, i.e., $p(\mathbb{X}(k)|\mathbb{Z}(k))$ using a discrete distribution comprising weighted samples (known as particles). To do this, N independent identically distributed particles, $\mathbb{X}^1, \cdots, \mathbb{X}^N$ are extracted from $p(\mathbb{X}(k)|\mathbb{Z}(k))$, and an empirical estimate of the distribution is constructed

$$\hat{p}(\mathbb{X}(k)|\mathbb{Z}(k)) = \frac{1}{N}\sum_{j=1}^{N} \delta_{\mathbb{X}^j(k)}(\mathbb{X}(k))$$

where $\delta_{\mathbb{X}^j(k)}$ denotes the Dirac mass at particle $\mathbb{X}^j(k)$. Then the expectation of any integrable function, g, can be estimated by,

$$\begin{aligned} E[g(\mathbb{X}(k),k)] &\approx \int g(\mathbb{X}(k),k)\hat{p}(\mathbb{X}(k)|\mathbb{Z}(k))d\mathbb{X}(k) \\ &= \frac{1}{N}\sum_{j=1}^{N} g(\mathbb{X}^j(k),k) \end{aligned}$$

This estimator is unbiased and (under weak assumptions) converges to the true expectation as the number of particles N tends to infinity, see e.g. [8]. In cases where it is not possible to sample from $p(\mathbb{X}(k)|\mathbb{Z}(k))$ directly, a technique known as importance sampling, can be employed. An algorithm to perform these operations recursively is Sequential Importance Resampling (SIR), see [10]. Particle Filter (PF) can deal with nonlinear models and non-gaussian noise. Moreover, the fact that targets do not appear in cameras field of views (FOV) can be used by PF in order to refine targets prediction. A potential problem of PF is the loss of exploration capabilities along the time. More details about how PF has been implemented for the surveillance problem can be found in [21]. Video [29] gives an example of the application of PF to the surveillance problem.

<u>Filter.C</u>

Extended Kalman Filter + Particle Filter

This section aims to explain the combined extended Kalman filter and particle filter approach. The goal is to try to combine the best features of the EKF and the PF. More specifically, the extended Kalman filter is used whenever the target is detected since it is much more robust to bad measurements. If the target is not detected, the particle filter is exploited because of its ability to include the information of *not* having detected the target. The theoretical basics of the individual parts (PF and EKF) have already been briefly introduced in the preceding sections, hence this part focuses on the implementation alone. Again the combined algorithm is visualized by a flowchart diagram, see Figure 4.7.
For the sake of clarity, some details of the individual algorithms shown in the previous flowcharts are omitted. An iteration starts at the $k = k + 1$ box. The first decision box distinguishes whether the target has been detected or not. If so the algorithm proceeds on the left side of the flowchart which is the EKF part, otherwise it continues on the right side which shows the PF part. Both sides are seperately described below.

EKF part: the target was detected. As a first step it is checked whether the target has also been detected in the step before (i.e. whether EKF has been used in the last step). If that is the case, the algorithm directly proceeds with the 'Prediction I' block where the prior estimate of the current state and the prior measurement covariance matrix is computed. Then it continues with the update step. If on the other hand the target was not detected and therefore the EKF not used in the last step, it has to be reinitialized using the information available from the particle filter. In order to do that, first the PF steps compensation, update and resample are performed. Then the (EKF) posterior estimate of the current state and its measurement covariance matrix are generated as follows

$$\hat{\mathbf{x}}_{k|k} = \frac{1}{N}\sum_{j=1}^{N}\hat{\mathbf{x}}_k^j$$

$$P_{k|k} = \frac{1}{N-1}\sum_{j=1}^{N}(\hat{\mathbf{x}}_k^j - \hat{\mathbf{x}}_{k|k})(\hat{\mathbf{x}}_k^j - \hat{\mathbf{x}}_{k|k})^T$$

i.e. they are approximated using the mean and the sample covariance matrix of the PF's posterior particles. In the flowchart those two quantities are indicated by $\bar{\mathbf{x}}_k^{PF}$ and $\hat{P}_k^{PF}$ respectively. The reason why first the PF is updated is that otherwise the normal approximation of the PF prior distribution would not be good since it usually has holes from the areas where the cameras were looking but did not detect the targets. Thus we rather approximate the PF's posterior distribution.
Then, analogically to the case a detected target in the EKF, the posterior estimate of the current state is updated using the measurement. By predicting this quantity one timestep ahead the output is computed and the iteration finishes.

PF part: in the case that the target was not detected, one distinguishes again whether the target was detected in the last timestep or not. If it was not detected this means that in the last timestep the particle filter was already used so the algorithm proceeds with the same steps as in the PF only case. If on the other hand the target was detected in timestep $k-1$, the particle filter has to be initialized. Note that there are basically two possibilities for doing this. One is to initialize the PF using $\hat{\mathbf{x}}_{k-1|k-1}$ and $P_{k-1|k-1}$, predicting it ahead using the EKF prediction and then sampling N particles from $N(\hat{\mathbf{x}}_{k|k-1},\, P_{k|k-1})$. The other is to sample from $N(\hat{\mathbf{x}}_{k-1|k-1},\, P_{k-1|k-1})$ and using the PF prediction to get $\{\hat{\mathbf{x}}_k^{*,comp,i}, 1/N\}$. Since we are able to use realistic accelerations in the PF prediction, we use the latter variant. Note that since we know the value of the last time interval, we obtain particles $\hat{\mathbf{x}}_k^{*,comp,i}$ already compensated for any possible time discrepancies. The algorithm continues with the standard update and resampling step, visualized in one box for reasons of clarity. The final operation is to predict the posterior $\hat{p}(\mathbf{x}_k|z_{1:k})$ one step ahead and then use it to maximize the probability of detecting the targets in the next time step.

A first choice in the "Application-based C4E co-design methodology" is which type of filtering technique is to be used in the Prediction task: Filter.A (Extended Kalman Filter) or Filter.B (Particle Filter) or Filter.C (Extended Kalman Filter+Particle Filter).

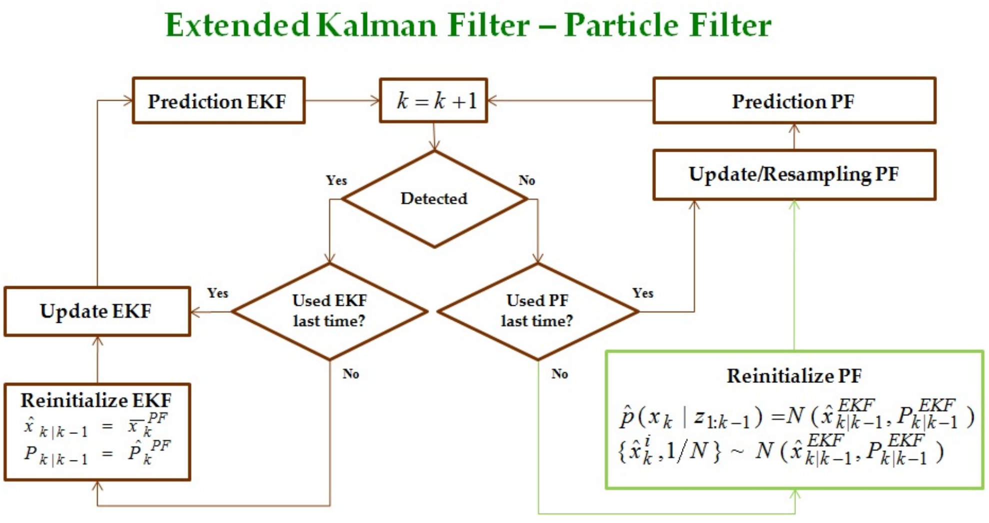
Extended Kalman Filter – Particle Filter
Prediction EKF
$k = k + 1$
Prediction PF
Update/Resampling PF
Yes
Detected
No
Yes
Update EKF
Used EKF last time?
Used PF last time?
Yes
No
No
Reinitialize EKF
$\hat{x}_{k|k-1} = \bar{x}_k^{PF}$
$P_{k|k-1} = \hat{P}_k^{PF}$
Reinitialize PF
$\hat{p}(x_k \mid z_{1:k-1}) = N(\hat{x}_{k|k-1}^{EKF}, P_{k|k-1}^{EKF})$
$\{\hat{x}_k^i, 1/N\} \sim N(\hat{x}_{k|k-1}^{EKF}, P_{k|k-1}^{EKF})$

Figure 4.7: EKF-PF flowchart

Computation of the Next Camera Position: Optimization

For the computation of the Next Camera Position, two different methods have been considered. The first method solves a one-step optimization problem where the objective is to minimize cameras movement and maximize target resolution while assuring a certain probability of keeping track of the object. The second method considers the problem of maximizing the probability of satisfying safety (tracking), reachability (target acquisition), and reach-avoid (one target tracking while acquiring another) on a time horizon N. The solution of the safety, reachability, and reach-avoid tasks are computed via dynamic programming resulting in an optimal control policy for the PTZ camera. This second method is still under investigation.

Method(1)

The objective of the tracking algorithm is to compute cameras inputs, i.e. zoom, pan and tilt angles, in order to guarantee a certain probability of targets detection at the next time instant while at the same time minimizing variation w.r.t. previous inputs and maximizing target resolution, i.e. "zoom in" value. This problem can be formulated as follows

$$\begin{array}{ll} \min\limits_{\theta_{i,k+1},\psi_{i,k+1},\zeta_{i,k+1}} & \sum_{i=1}^{N_{cam}} \kappa_1 \Delta\theta_i^2 + \kappa_2 \Delta\psi_i^2 + \kappa_3 \Delta\zeta_i^2 + \frac{\kappa_4}{\zeta_{i,k+1}^2} \\ \text{subject to} & |\theta_{i,k+1} - \theta_{i,k}| < \Delta\Theta_{max} \\ & |\psi_{i,k+1} - \psi_{i,k}| < \Delta\psi_{max} \\ & |\zeta_{i,k+1} - \zeta_{i,k}| < \Delta\zeta_{max} \\ & \Theta_{min} \leq \Theta_{i,k+1} \leq \Theta_{max} \\ & \psi_{min} \leq \psi_{i,k+1} \leq \psi_{max} \\ & \zeta_{min} \leq \zeta_{i,k+1} \leq \zeta_{max} \\ & Pr(target_j \in FOV_{\cup,k+1}) >= \alpha \\ & \\ & \forall i = 1, \dots, N_{cam}, \forall j = 1, \dots, N_{targ} \end{array} \tag{8}$$

where N_{cam} and N_{targ} are the number of cameras and targets respectively, $\kappa_1, \kappa_2, \kappa_3, \kappa_4$ are constants and $FOV_{\cup,k+1}$ is the union of the fields of view of all cameras at time $k+1$. The field of view of a camera is computed by projecting the outermost pixels on the CCD chip to the ground floor. When all of the individual fields of view are available it is straightforward to compute $FOV_{\cup,k+1}$.
The first three terms of the cost function penalize the deviation from the current position whereas the fourth penalizes a high 'zooming out'. The first three constraints are the dynamic and the second three are the positional constraints on the cameras. $Pr(target_j \in FOV_{\cup,k+1}) >= \alpha$ finally is the constraint that the probability of target j being inside the combined field of view at the next time step must be greater than α.

Constraint.A

When EKF is exploited, the probability of detection constraint is assured by using the method described in [20], that numerically approximates the integral of a bivariate normal distribution over an arbitrary polygon.

Constraint.B

When PF is used, the probability of detection is assured by requiring at least a percentage α of particles to be inside cameras FOV at the next time.

If Method(1) is considered, a choice in the "Application-based C4E co-design methodology" regards the computation of $Pr(target_j \in FOV_{\cup,k+1}) >= \alpha$, *which is part of the task Computation of the Next Camera Position. Two different solutions have been proposed Constraint.A (numerically approximation of the integral of a bivariate normal distribution over a polygon) or Constraint.B (count the percentage of particles inside the field of view).*

Optimization.A

Note that the optimization problem (8) is nonlinear because the change in the field of view is not linear in pan, tilt and zoom and nonconvex because the approximated pdf of the particle filter is arbitrary in the general case. Therefore very general algorithms have to be applied, at the usual cost of slow convergence and subobtimality. One possible solution for the optimization problem consists in to solve the problem by discretizing the search space, enumerating all discretized positions and choosing the best one. The search space of each camera was discretized into $N_\Theta \cdot N_\psi \cdot N_\zeta$ points, where N_Θ, N_ψ and N_ζ indicate the resolution of the discretization in pan, tilt and zoom direction, respectively.

Optimization.B

The optimization is performed by using Simulated Annealing (SA), see e.g. [14], with time limit equal to the sampling time.

Optimization.C

One of the major drawbacks of simulated annealing is its very slow convergence, especially for problems with large search space. Under the assumption that each camera has a dedicated processor on board and communication between cameras is available, a parallel versions of SA can be implemented. We assume also that cameras are calibrated, i.e. their homographies are known to all the cameras. There have been many attempts to develop parallel versions of simulated annealing, see e.g. [1]. One of the approaches described in [1] is the clustering algorithm. Such algorithm takes advantage of the fact that a good initial solution provides SA faster convergence. Initially, the n cameras of the network run SA algorithm using different initial solutions. It is assumed that processors have different number seeds. After a fixed number of iterations, they exchange their partial results. The best partial solution is then used as new initial solution for all processes and SA is started again. This process is repeated until time limit (i.e. sampling time) is exceeded.

When the number of targets and cameras involved in the problem is big, the SA search space is very large. This means that even finding a feasible solution to problem (8) will be difficult, especially with a time limit constraint. For this reason, one can think about solving problems with a smaller number of optimization variables by fixing some cameras inputs to their previous values, i.e. that cameras will not be moved. Given n, number of cameras, and n_a number of cameras we want to move, we can solve in parallel $\frac{n!}{a!(n-a)!}$ (all the possible combinations without repetitions) problems on different processors, i.e. cameras, and then, by communicating, choose the minimum cost feasible solution. A possible strategy could be the following:

- start the procedure with $a = 1$
- solve in parallel all possible combinations $\binom{n}{a}$ for a fixed number of iterations
- initialize all $\binom{n}{a}$ SA optimization problems with the minimum cost feasible solution of $a - 1$
- iterate the procedure till time limit is exceeded.

Suppose for example $n = 3$. When $a = 1$ we solve on the three processors in parallel the problems with active only camera 1 or 2 or 3. Then for $a = 2$, we solve in parallel the problems with active cameras $1-2$, $1-3$, $2-3$. For $a = 3$ we have just to solve the problem with all the cameras active. In that case the clustering algorithm can be employed for parallelizing the problem and speeding up the convergence.

Another expedient for reducing the number of optimization variables is to exclude a priori from the optimization the cameras that can not really contribute in increasing target probability of detection. This evaluation can be easily done by looking at cameras constraint.

If Method(1) is considered, a choice in the "Application-based C4E co-design methodology" regards the way we solve problem (8). Two different solutions have been proposed: Optimization.A (discretization of the search space and exhaustive search of the optimal solution) or Optimization.B (simulated annealing) or Optimization.C (distribution of Optimization.B).

Method(2)

Let n adversary objects be distributed in a general spatial frame $X_G \subset \mathbb{R}^3$ (e.g. an auditorium or a stadium). Each set-valued evader $\mathcal{O}^{(i)} \in \mathcal{B}(X_G)$, $i \in \{1, ..., n\}$ ($\mathcal{B}(\cdot)$ denotes the Borel $\sigma-$algebra.), is parameterized by a set of parameters $x_e \in X_e$, where X_e is the state space of the adversary parameterization. The evader set is constrained in X_G and the parametric representation constrained in X_e. We assume that the set-valued evaders are independent and can intersect.

We consider a parametric model for the camera with state x and state space X, $x \in X$. Naturally, there exists a mapping from the state of the camera parameterization x to the set-valued camera view in the spatial frame X_G (defined as the field of view (FOV)). In the general case, this function can be defined as a measureable mapping $\mathcal{L} : X \rightarrow \mathcal{B}(X_G)$.

In the spatial frame X_G, the set

$$S_1 = \{x_G \in X_G : x_G \in \cup_i \mathcal{O}^{(i)}\}$$

comprises all states that intersect with the set-valued region of one or more evaders (equivalently the union of the evader sets). S_1 can be seen as the coverage of the evaders in the spatial frame.

In the camera space X, the set

$$S_2 = \{x \in X : \mathcal{L}(x) \cap \mathcal{O}^{(i)} \neq \emptyset, \;\; \forall i \in \{1, ..., n\}\}$$

comprises all camera states for which every evader is in the field of view of the camera. Likewise, assuming $n_1 < n$,the set

$$S_3 = \{x \in X : \mathcal{L}(x) \cap \mathcal{O}^{(i)} \neq \emptyset, \;\; \forall i \in \{1, ..., n_1\}\}$$

comprises all camera states for which n_1 evaders are in the view of the camera. Lastly, the set

$$S_4 = \{x \in X : \exists i \in \{1, ..., n\}, \;\; \mathcal{L}(x) \cap \mathcal{O}^{(i)} = \emptyset\}$$

comprises all camera states for which one or more evaders is not visible by the camera).

It is often the case in visual surveillance that the state of the evader is not exactly known. Further, it is almost always impossible to deterministically predict the path that an evader will take in the future. Thus, as uncertainty plays a large role in the estimation and prediction of evader trajectories, dealing with this uncertainty is central to the success of an automated surveillance system. Consider, for instance, in the current example that the parametrization of each set-valued evader $i \in \{1, ..., n\}$ is distributed according to some probability distribution. Under this consideration, it is no longer just a question of whether an evader (or set of evaders) is visible to the camera when the camera is in state x. Rather, the question becomes the following: What is the probability that the evader (or set of evaders) is visible to the camera when the camera is in state x?

It follows that S_1, S_2, S_3, and S_4 are random sets according to the random distribution of the evader parametrization considered above. Further, note that they are dependent random sets (they are parameterized by the random evader centers) and that $S_2 \subseteq S_3$ almost surely. Thus, the question above can be answered by computing the covering functions of the various sets. Specifically, the covering function $p_{S_1}(x_G)$ defines the probability that $x_G \in X_G$ will intersect with one or more evaders. Similarly, $p_{S_2}(x)$ and $p_{S_3}(x)$ represent the probability that the camera in state $x \in X$ will visually capture evaders $\{1, ..., n\}$ and $\{1, ..., n_1\}$ respectively.

Considering the probabilistic sets detailed above, it is possible to formulate surveillance tasks using the theory of stochastic reachability for DTSHS [23, 25] and stochastic reachability with random sets [22, 24] upon which the framework for autonomous surveillance is built. See [12] for details.

Safety(tracking): Minimize the probability that the camera loses sight of one of the evaders at some point during the time horizon $k \in \{0, ..., N\}$. It follows that the safety problem can be formulated where the camera state space X denotes the safe set and the set S_4 denotes the target set and the optimal control policy is obtained by solving the DP of Theorem 6 in [12] in the minimal case (i.e. the sup is replaced by inf).

Reach(acquisition): Maximize the probability that the camera can reach all n evaders at some point during the finite time horizon $k \in \{0, ..., N\}$. It follows that the reach problem can be formulated where the camera state space X denotes the safe set and the set S_2 denotes the target set and the optimal control policy is obtained by solving the DP of Theorem 6 in [12].

Reach-Avoid: Maximize the probability that the camera can reach all n evaders at some point during the finite time horizon $k \in \{0, ..., N\}$ while avoiding losing a subset of evaders at each prior time point. It follows that the reach-avoid problem can be formulated where the set S_3 denotes the safe set and the set S_2 denotes the target set and the optimal control policy is obtained by solving the DP of Theorem 6 in [12].

For computational purposes, the DPs have been solved by discretizing the environment X_G and the cameras space X.

A choice in the "Application-based C4E co-design methodology" regards the method we use for solving the tracking problem. Two different methods have *been proposed. Method(1) and Method(2).*

Iterations

In the implementation of the "Application-based C4E co-design methodology", we started by selecting a possible solution to the surveillance problem. We then refined our choices in order to better address the problem requirement. Initially we started from the control selecting 250 ms as sampling time, Filter.A, Method(1), Constraint.A and Optimization.A. Such solution has the advantage of being simple. However, it has some limitations. First of all, even if Filter.A is computationally efficient, it gives mediocre performance and can be used when targets are slow. Qualitatively, see Figure 4.8, in order to address the minimum control quality requirements, this method requires a number of agents that makes the problem unfeasible, i.e. computational constraints are not satisfied.

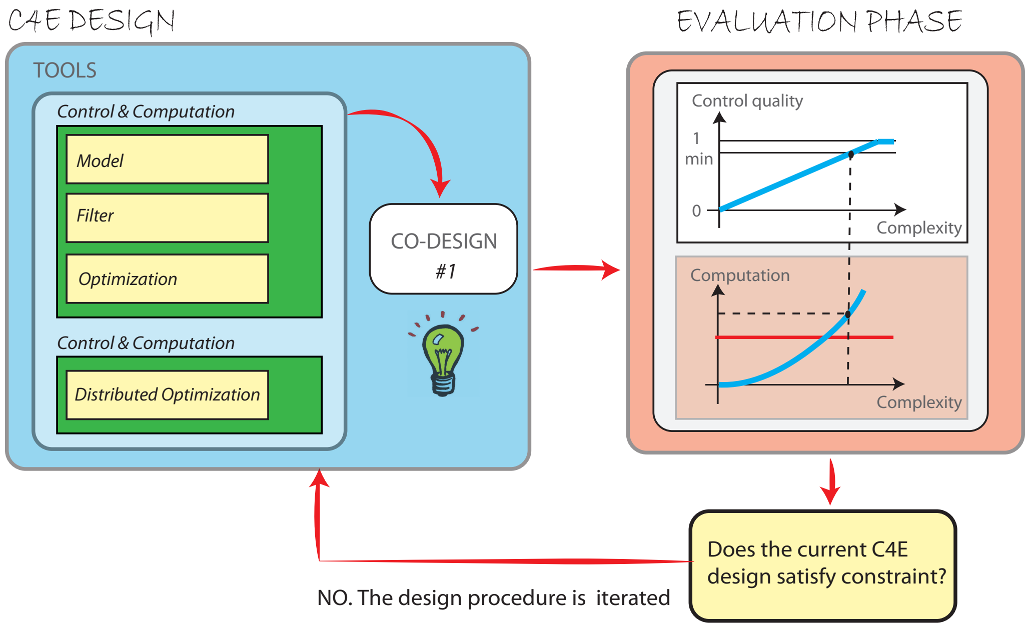

Figure 4.8: Co-design for smart networks of cameras: co-design 1.

For such reasons, Filter.B+Constraint.B have been considered. The advantages are: not detected information can be included in the filter and we can get more realistic prediction (due to the non-normal noise). On the other side PF suffer of the problem of Particle impoverishment and can be used only when the detection is very good. For these reasons we then explored

Filter.C+Constraint.A+Constraint.B. The EKF-PF prediction scheme combines the strengths of Filter.A and Filter.B. It takes advantage of the robust incorporation of the measurements of the EKF and also of the ability of including the information that the target was not inside the field of views. The only downside is that the EKF-PF does not use the superior prediction of the particle filter in the case that the targets are detected. A slow-motion (x0.5) video showing the principle of EKF-PF scheme is uploaded to [30]. We then had to consider the optimization problem. If the number of cameras is limited to 2, Optimization.A gives good performance. On the other side, if the number of cameras increases, such solution is not feasible anymore. In that case, Optimization.B is employed. In order to distribute the optimization problem and to deal with more complex scenarios, Optimization.C is a possible solution. Qualitatively, see Figure 4.9, with this second method we achieve the minimum desired control quality with less agents. On the other side the method is still computationally too expensive.

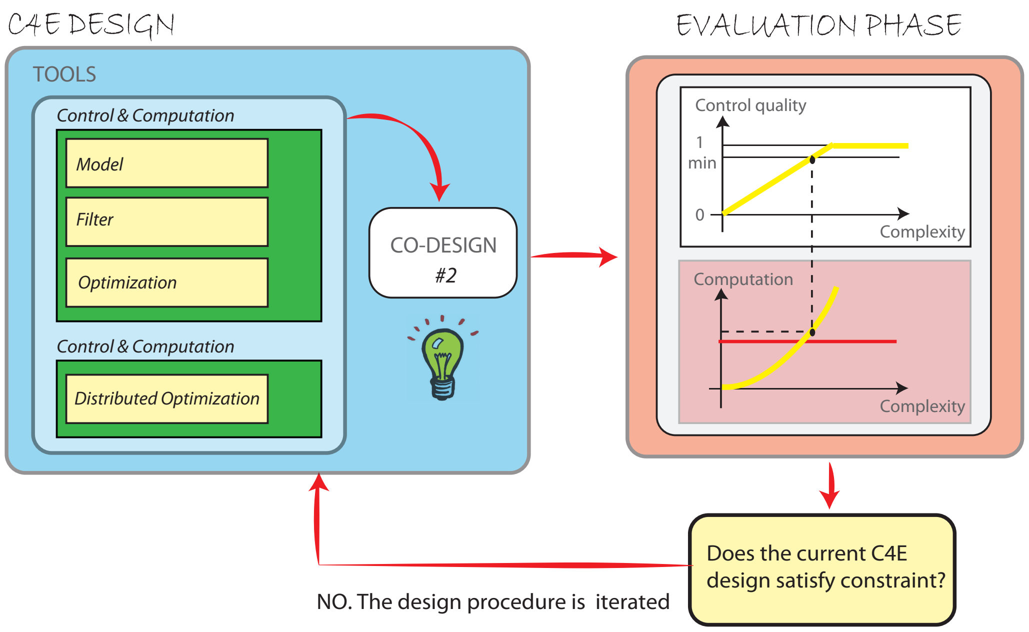

Figure 4.9: Co-design for smart networks of cameras: co-design 2.

We are currently working on Method(2) because we believe it can provide better performance than Method(1). We are currently able with this method to solve an optimization problem over an horizon longer than one and the solution of the DP is very fast. The combination of Filter and Method(2) is still under investigation. We believe that the value function, output of the

DP, could be used to take higher level decisions like tasks allocation and to decentralize/distribute the tracking problem. With this method we expect to be able to reduce the complexity needed to address the performance requirements while at the same time respecting the computational constraints, see Figure 4.10.

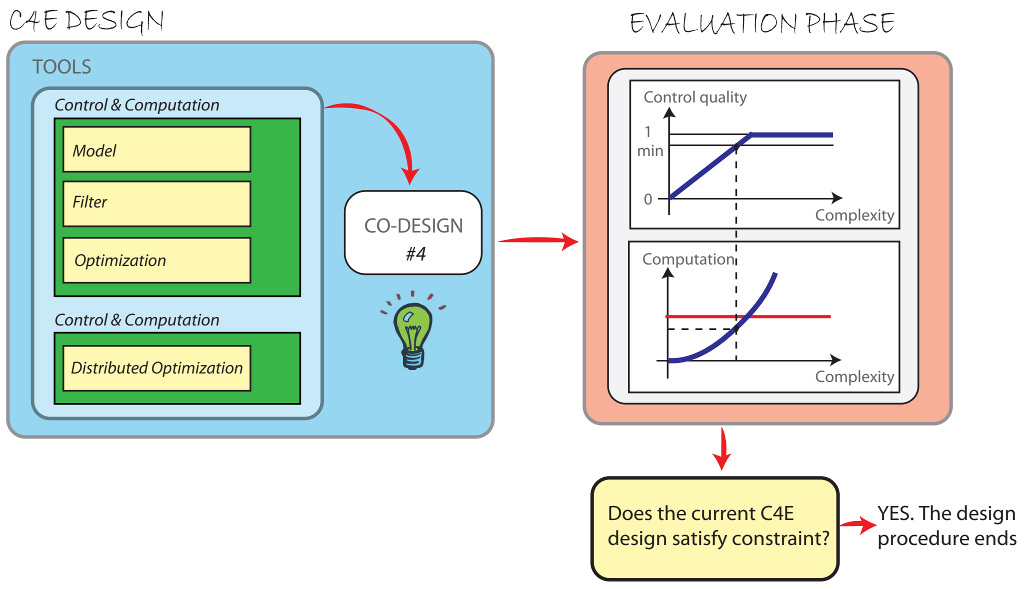

Figure 4.10: Co-design for smart networks of cameras: co-design 3.

The choices we made so far clearly involve Control, Computation and Complexity. The objective is to provide a method able to give good tracking performance and scalability. Method(2) seems very promising and deserves further investigation. The application of such method to the case of a network of cameras for motion captures seems also possible and we are currently working on this extension of the framework.

4.2 Co-design framework for the fleet of AUVs

4.2.1 Phase I: Specifications

As we discussed in Section 3.3, the communication between the underwater vehicles is subject to short range communication, noise and known transmission delays. Also, the ASV on board computer, which runs the high level control law may be subject to limited computation constraints. In addition, each AUV is powered by batteries with limited life time.

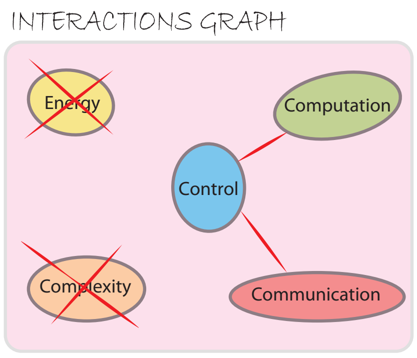

Figure 4.11: Summary of interactions between control, communication, and computation in the fleet of AUVs

4.2.2 Phase II: Interactions graph

As mentioned in the previous chapter, the objective of the fleet of AUVs is to locate the source of an underwater concentration flow by moving towards the source. To achieve this goal, each AUV senses concentration flow locally; and this information is used to move the fleet towards the source. In this fleet, navigation and concentration data are exchanged between vehicles via acoustic waves; but this communication is subject to imperfections. Moreover, this fleet may be subject to computational limitations; in particular, if all AUVs is coordinated by ASV. Therefore, in this fleet, we have interactions between control, communication, and computation. Figure 4.11 summarizes the interactions between different C4E components for this case study. This represents Phase II of the co-design framework outlined in the previous chapter. Note that since above technique involves 3 to 10 AUVs; hence complexity of algorithms is not an issue.

4.2.3 Phase III: Co-Design Iterations

As mentioned, the objective of the fleet is to move towards the source as fast as possible. Therefore, control is the most important component of the co-design framework. As discussed earlier, our contribution so far, for the fleet of AUVs, is a cooperative fleet control law [15], which is responsible for the formation movement. As shown in Fig. 4.12, the technique presented in [15] forms a circular formation of AUVs with a time varying center, which follows the gradient of the concentration flow and reaches eventually to the source. As shown in Fig. 4.13, this technique involves a two - level control: Low level control (or local controller per each AUV) and high level control

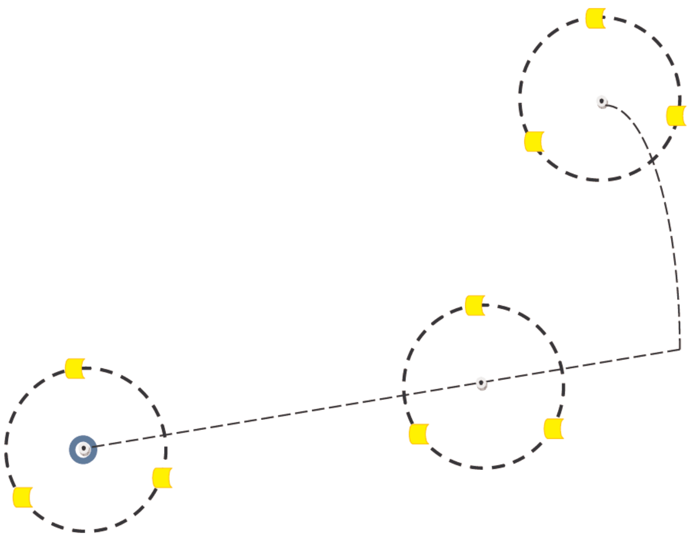
Figure 4.12: Fleet of AUVs

located at ASV.

The low level control is responsible for the vehicle formation. Each AUV sends its sampled concentration data via wireless link to the distant ASV. Then, the ASV on-board computer reconstructs/estimates the sampled concentration data and subsequently, updates the reference for movement and formation based on the estimated values. Due to long transmission delays in exchanging data between ASV and AUVs, the available time for processing the received information from AUVs and updating the reference trajectory by the ASV on-board computer is limited. Besides, the ASV on-board computer is responsible for other tasks. Therefore, the high level control law, which is responsible for updating the reference trajectory, must be developed in the presence of limited computational resource. In addition, exchanging data via wireless links is subject to communication imperfections (e.g., channel noise). While the communication from ASV to all AUVs can be considered noiseless (this transmission can be done via high power wireless communication), the transmission from each AUV to ASV is subject to random packet dropout.

The control loop between each AUV and ASV of Fig. 4.13 is shown in Fig. 4.14, which is subject to random packet dropout. In the absence of computation constraint and communication imperfections, the policy presented in [15] results in a satisfactory performance. Unfortunately, this is not the

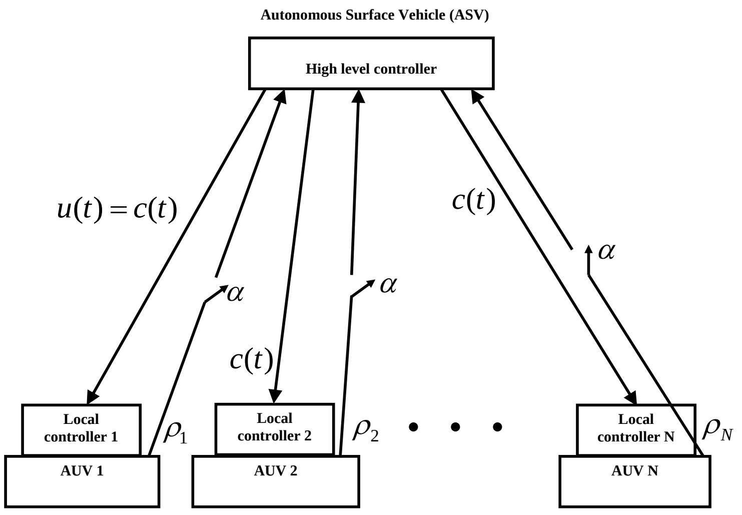

Figure 4.13: Exchanging the sampled concentration data and the reference center between AUVs and ASV

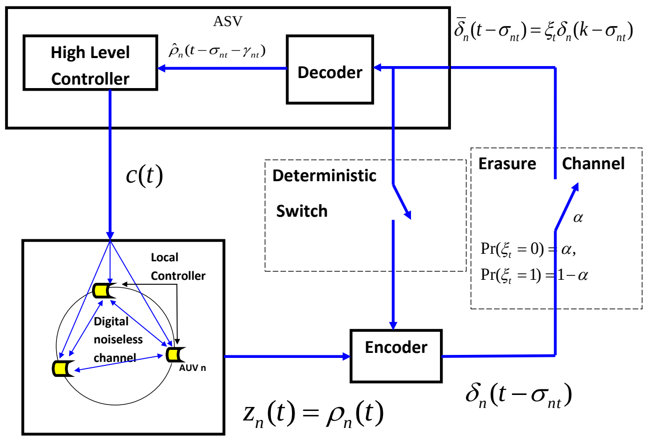

Figure 4.14: Control loop between each AUV and ASV

case when there are the effects of communication imperfections. Therefore, to compensate the effects of communication imperfections (random packet dropout), we need to include proper encoder and decoder in the closed loop feedback system (See Fig. 4.14). In developing the coding scheme, we can use feedback channel. When a packet is received successfully, we can send back an acknowledgment bit to indicate the packet was received successfully. Then, the transmitter can use this information in preparing the next packets. But, we can not use feedback channel all the time. Because, it results in a long search mission. This is due to long transmission delay in exchanging information between underwater vehicles, which is caused by slow propagation speed of underwater sound waves. On the other hand, if we do not use feedback channel at all, as shown in [7], we will end up with a coding scheme with high computational complexity. By coding computational complexity we mean the maximum of times, which are spent by encoder plus decoder in each sampling period, to have a pre-defined reliable communication. Hence, if we use a coding scheme with high computational complexity, we need to increase the time period between transmission of two successive sampled concentration data (i.e., the TDMA time slot) to avoid computational overflow. But, long TDMA time slots result in a poor performance (e.g., long search mission). Therefore, as shown in Fig. 4.15 and Fig. 4.16 in the fleet of AUVs, it is natural to look at the trade - off between duty cycle for feedback channel use, which is denoted by β ($0 \leq \beta \leq 1$, $\beta = 0$ corresponds to the case of no feedback channel at all, and $\beta = 1$ corresponds to the case of feedback channel all the time), coding computational complexity, and performance, which is represented by the duration for the search mission. In fact, a proper selection of β (denoted by β_{opt}) will result in a suitable coding computational complexity; and therefore, a desirable duration for the search (i.e., the smallest duration for the search, T_{min}).

To find out this trade - off, we suggest the following steps:

Step 1: Obtaining a dynamical model for the sampled concentration data.

Step 2: Designing a coding scheme, which results in a real time reliable communication, for a given β.

Step 3: Calculation of the coding computational complexity and duration of search for this β.

By choosing a different β and repeating Steps 2-3, we can obtain Fig. 4.15 and Fig. 4.16; and subsequently, β_{opt} and T_{min}.

In the Appendix I, we present a dynamic model for sampled concentration flow. We also present some of available coding techniques for $\beta = 0$ and $\beta = 1$.

Iterating the design procedure above, if the minimum duration for the search, T_{min}, obtained from the previous phase, is less than AUVs' life time,

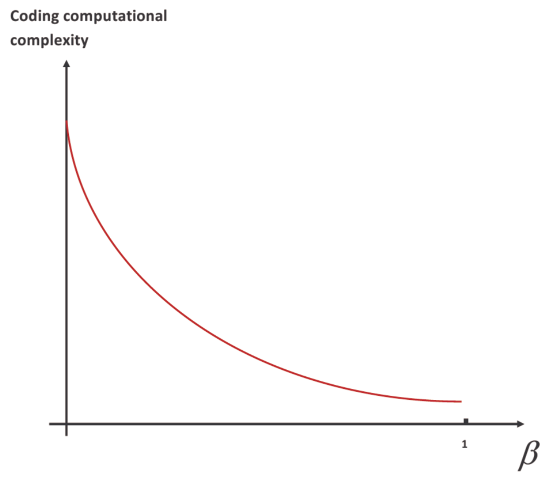

Figure 4.15: Trade - off between β and coding computational complexity

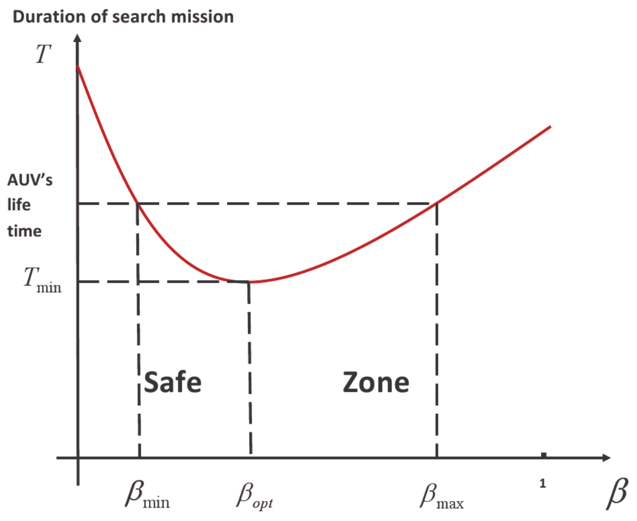

Figure 4.16: Trade - off between β and the quality of search

we have accomplished our design goal. Otherwise, we need to find a different control strategy and repeat the Phase III.

5 Conclusions

We proposed in this report a novel C4E co-design methodology that allows the integration of control-estimation, communication, computation, complexity and energy considerations within a unified framework. This co-design methodology comprises three phases:

1. *Design specifications*: the problem is formalized and all the design requirements are outlined.

2. *C4E interactions graph*: the five elements of the C4E are considered as the nodes of a graph and the effects of one element of all the others are represented by directed edges.

3. *Co-Design iterations*: given the design specifications and the interaction graph, the designer proposes a first co-design. An evaluation phase quantifies the performance of the proposed approach. The evaluation phase provides feedback to the C4E design specifying which design components have to be modified. The C4E design is iterated until objectives are addressed.

We illustrated how this novel co-design methodology can be utilized within the case studies of FeedNetBack in order to improve the overall system performance. In particular, we considered the following two case studies:

- *Surveillance systems using a network of smart cameras*

- *A fleet of Autonomous Underwater Vehicles (AUVs)*

The constraints imposed by control, communication, computation, complexity, and energy have been analyzed and several co-design iterations have been performed that resulted in an improved system performance. In this co-design process, we relied on several of the design methods involving a single or at most two of the elements of the C4E, which have been developed within WP2-5.

References

[1] E. Aarts and J. Korst. *Simulated annealing and Boltzmann machines.* John Wiley & Sons New York, 1989.

[2] L. B. Arranz, A. Seuret, and C. Canudas de Wit. Translation control of a fleet circular formation of auvs under finite communication range. *The 48th IEEE Conference on Decision and Control*, pages 8345–8350, 2009.

[3] Alberto Bemporad, Manfred Morari, Vivek Dua, and Efstratios N. Pistikopoulos. The explicit linear quadratic regulator for constrained systems. *Automatica*, 38(1):3 – 20, 2002.

[4] P. Billingsley. *Probability And Measure.* John Wiley and Sons, 1995.

[5] A. Cenedese, F. Cerruti, M. Fabbro, C. Masiero, and L. Schenato. Decentralized task assignment in camera networks. *Conference on Decision and Control*, 2010. accepted.

[6] M. Chiodo, P. Giusto, A. Jurecska, H.C. Hsieh, A. Sangiovanni-Vincentelli, and L. Lavagno. Hardware-software codesign of embedded systems. *Micro, IEEE*, 14(4):26–36, 1994.

[7] G. Como, F. Fagnani, and S. Zampeiri. Anytime reliable transmission of real - valued information through digital noisy channels. *SIAM J. Control Optim.*, 48:3903–3924, 2010.

[8] D. Crisan and A. Doucet. A survey of convergence results on particle filtering methods for practitioners. *IEEE Transactions on signal processing*, 50(3):736–746, 2002.

[9] A. Doucet, N. De Freitas, and N. Gordon. *Sequential Monte Carlo methods in practice.* Springer Verlag, 2001.

[10] A. Doucet, S. Godsill, and C. Andrieu. On sequential Monte Carlo sampling methods for Bayesian filtering. *Statistics and computing*, 10(3):197–208, 2000.

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

[12] N. Kariotoglou, D.M. Raimondo, S. Summers, and J. Lygeros. A stochastic reachability framework for autonomous surveillance with pan-tilt-zoom cameras. In *Proc. of the 50th IEEE Conf. on Decision and Control, submitted*, Orlando, Florida, 2011.

[13] K. Keutzer, A.R. Newton, J.M. Rabaey, and A. Sangiovanni-Vincentelli. System-level design: Orthogonalization of concerns and platform-based design. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 19(12):1523–1543, 2000.

[14] S. Kirkpatrick. Optimization by simulated annealing: Quantitative studies. *Journal of Statistical Physics*, 34(5):975–986, 1984.

[15] B. J. Moore and C. Canudas de Wit. Source seeking via collaborative measurements by a circular formation of agents. *American Control Conference*, 2010.

[16] G. N. Nair, F. Fagnani, S. Zampieri, and R. J. Evans. Feedback control under data rate constraints: An overview. *Proceedings of the IEEE*, 95(1):108–137, 2007.

[17] S.I. Niculescu. *Delay effects on stability: A robust control approach*. Springer Verlag, 2001.

[18] D. M. Raimondo, S. Gasparella, D. Sturzenegger, J. Lygeros, and M. Morari. A tracking algorithm for ptz cameras. *2nd IFAC Workshop on Distributed Estimation and Control in Networked Systems*, Annecy, France, 2010.

[19] L. Schenato, B. Sinopoli, M. Franceschetti, K. Poolla, and S. S. Sastry. Foundations of control and estimation over lossy networks. *Proceedings of the IEEE*, 95:163–187, 2007.

[20] A. Sommariva and M. Vianello. Product Gauss cubature over polygons based on Green's integration formula. *BIT Numerical Mathematics*, 47(2):441–453, 2007.

[21] D. Sturzenegger. Multi target tracking using multiple pan-tilt-zoom cameras. Master's thesis, Automatic Control Laboratory, ETH Zurich, 2010.

[22] S. Summers, M. Kamgarpour, C.J. Tomlin, and J. Lygeros. A Stochastic Reach-Avoid Problem with Random Obstacles. In *HSCC (Hybrid Systems: Computation and Control)*, Chicago, USA, April 2011.

[23] S. Summers and J. Lygeros. A Probabilistic Reach-Avoid Problem for Controlled Discrete Time Stochastic Hybrid Systems. In *IFAC Conference on Analysis and Design of Hybrid Systems, ADHS*, Zaragoza, Spain, September 2009.

[24] Sean Summers, Maryam Kamgarpour, Claire Tomlin, and John Lygeros. A stochastic reach-avoid decision problem with random sets. 2011. Submitted.

[25] Sean Summers and John Lygeros. Verification of discrete time stochastic hybrid systems: A stochastic reach-avoid decision problem. *Automatica*, 46(12):1951 – 1961, 2010.

[26] S. Tatikonda and S. Mitter. Control under communication constraints. *Automatic Control, IEEE Transactions on*, 49(7):1056–1068, 2004.

[27] J. Tsitsiklis, D. Bertsekas, and M. Athans. Distributed asynchronous deterministic and stochastic gradient optimization algorithms. *Automatic Control, IEEE Transactions on*, 31(9):803–812, 2002.

[28] Video1, 2010. `http://control.ee.ethz.ch/~rdavide/videos/EKF_principle3_0x0p5.mp4`.

[29] Video2, 2010. `http://control.ee.ethz.ch/~rdavide/videos//PF_principle2_0x0p5.mp4`.

[30] Video3, 2010. `http://control.ee.ethz.ch/~rdavide/videos/EKFPF_principle2_0x0p5.mp4`.

Appendix I

Modeling sampled concentration data

In the stationary regime, the diffusion law gives an exponential decreasing spatial distribution of concentration, represented in 2D by the following equation:

$$\rho(x,y) = \alpha e^{-K\left((x-x_s)^2+(y-y_s)^2\right)}, \tag{9}$$

where (x_s, y_s) is the center of the source, K the distribution spread, and α the scaling coefficient. Note that the parameters α, K, and (x_s, y_s) are not known to ASV a priori.
Most of the time, above concentration distribution is distorted, e.g., due to waves. To represent this effect, above distribution is replaced by the summation of three concentric ellipses , as follows:

$$\begin{aligned}\rho(x,y) &= \alpha_1 e^{-\left(a_{11}(x-x_s)^2+a_{12}(x-x_s)(y-y_s)+a_{13}(y-y_s)^2\right)}\\ &+\alpha_2 e^{-\left(a_{21}(x-x_s)^2+a_{22}(x-x_s)(y-y_s)+a_{23}(y-y_s)^2\right)}\\ &+\alpha_3 e^{-\left(a_{31}(x-x_s)^2+a_{32}(x-x_s)(y-y_s)+a_{33}(y-y_s)^2\right)},\end{aligned}$$

where $a_{11}a_{13} - \frac{a_{12}^2}{4} > 0$, $a_{21}a_{23} - \frac{a_{22}^2}{4} > 0$, $a_{31}a_{33} - \frac{a_{32}^2}{4} > 0$, parameters a_{ij}, α_i with $i, j \in \{1, 2, 3\}$ are unknown and subject to change, and the pair (x_s, y_s) is unknown but fixed.
For simplicity, in what follows we assume that the environment is calm enough such that the concentration distribution in 2D space is represented by the following elliptic model:

$$\rho(x,y) = \alpha e^{-\left(a(x-x_s)^2+b(x-x_s)(y-y_s)+c(y-y_s)^2\right)}, \tag{10}$$

where $ac - \frac{b^2}{4} > 0$, the concentration parameters α, a, b, and c are unknown and subject to slow change, and the pair (x_s, y_s) is unknown and fixed.
In the beginning of the search mission, AUVs form a uniformly distributed circular formation with center C_d and radius R. Arranz, Seuret, and Canudas de Wit in [2] presented policies for local controllers that result in uniformly distributed circular formation with center C_d and radius R. Now, consider the nth AUV $(3 \leq n \leq N \leq 10)$ on a uniformly distributed circular formation of AUVs with radius R and center C_d, as shown in Fig. 5.1. Let the vector

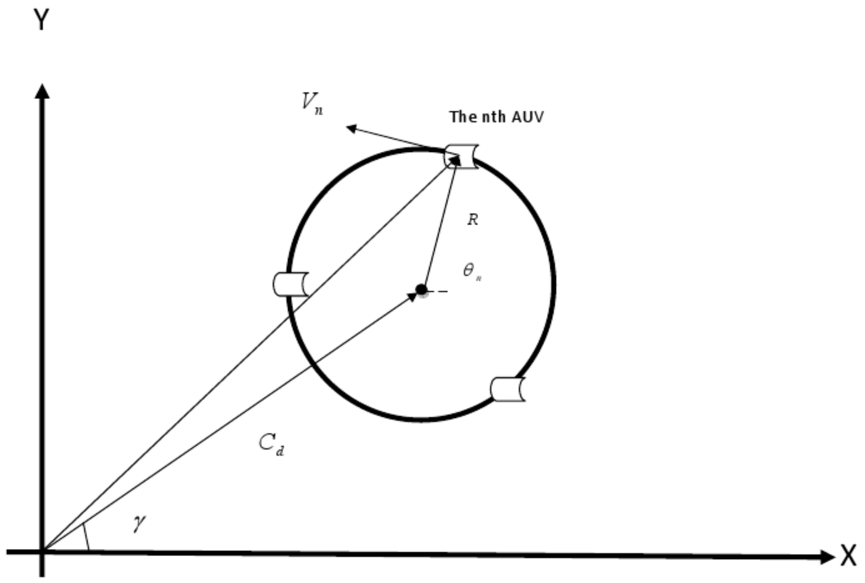

Figure 5.1: Illustration of an AUV on a circular formation with center C_d and radius R

V_n (m/s) denote the vehicle's forward velocity and $\theta_n(t) = \omega t$ denote the heading angle, where ω (rad/s) is the rotational speed (See Fig. 5.1). Note that $|V_n|$, R, ω, C_d, and γ are known to ASV a priori, where $|\cdot|$ denotes the magnitude. It is easy to verify that

$$\begin{pmatrix} x_n(t) \\ y_n(t) \end{pmatrix} = \begin{pmatrix} C_d cos\gamma \\ C_d sin\gamma \end{pmatrix} + \begin{pmatrix} Rcos(\omega t + \frac{2\pi}{N}n) \\ Rsin(\omega t + \frac{2\pi}{N}n) \end{pmatrix}. \tag{11}$$

$$\begin{pmatrix} \dot{x}_n(t) \\ \dot{y}_n(t) \end{pmatrix} = \begin{pmatrix} -|V_n|sin(\omega t + \frac{2\pi}{N}n) \\ |V_n|cos(\omega t + \frac{2\pi}{N}n) \end{pmatrix}. \tag{12}$$

Let $\rho(x_n(t), y_n(t))$ be the sampled concentration data provided by the nth AUV at point $(x_n(t), y_n(t))$. Then,

$$\begin{aligned} \dot{\rho}(x_n(t), y_n(t)) = \frac{d\rho(x_n(t), y_n(t))}{dt} &= \frac{\partial\rho(x_n(t), y_n(t))}{\partial x_n(t)}\frac{dx_n(t)}{dt} \\ &\quad + \frac{\partial\rho(x_n(t), y_n(t))}{\partial y_n(t)}\frac{dy_n(t)}{dt}. \end{aligned} \tag{13}$$

Note that above equation also involves the following terms:

$$\frac{\partial\rho(x_n(t), y_n(t))}{\partial i}\frac{di}{dt} \qquad \text{with} \quad i \in \{\alpha, a, b, c, d\}$$

But, as $\frac{\partial \rho(x_n(t),y_n(t))}{\partial i}$ are bounded and the concentration parameters α, a, b, c are changing slowly during AUVs maneuver on a given circular formation, we can assume that these terms are negligible.
From (10), (11), and (13), it follows that:

$$\begin{aligned}
&\dot{\rho}(x_n(t), y_n(t)) \\
&= [-\Big(2a(x_n(t) - x_s) + b(y_n(t) - y_s)\Big)\dot{x}_n(t) \\
&-\Big(b(x_n(t) - x_s) + 2c(y_n(t) - y_s)\Big)\dot{y}_n(t)]\rho(x_n(t), y_n(t)) \\
= \ &[\Big(2a(C_d cos\gamma + Rcos(\omega t + \frac{2\pi}{N}n) - x_s) \\
&+b(C_d sin\gamma + Rsin(\omega t + \frac{2\pi}{N}n) - y_s)\Big)|V_n|sin(\omega t + \frac{2\pi}{N}n) \\
&-\Big(b(C_d cos\gamma + Rcos(\omega t + \frac{2\pi}{N}n) - x_s) \\
&+2c(C_d sin\gamma + Rsin(\omega t + \frac{2\pi}{N}n) - y_s)\Big)|V_n|cos(\omega t + \frac{2\pi}{N}n)]\rho(x_n(t), y_n(t)).
\end{aligned}$$

For the simplicity of notation, let us denote $\rho(x_n(t), y_n(t))$ by $\rho_n(t)$. Then, the discrete version of the above linear time varying continuous dynamic system is:

$$\begin{aligned}
&\frac{\rho_n(t+1) - \rho_n(t)}{T} \\
= \ &[\Big(2a(C_d cos\gamma + Rcos(\omega Tt + \frac{2\pi}{N}n) - x_s) \\
&+b(C_d sin\gamma + Rsin(\omega Tt + \frac{2\pi}{N}n) - y_s)\Big)|V_n|sin(\omega Tt + \frac{2\pi}{N}n) \\
&-\Big(b(C_d cos\gamma + Rcos(\omega Tt + \frac{2\pi}{N}n) - x_s) \\
&+2c(C_d sin\gamma + Rsin(\omega Tt + \frac{2\pi}{N}n) - y_s)\Big)|V_n|cos(\omega Tt + \frac{2\pi}{N}n)]\rho_n(t),
\end{aligned}$$

where T is the sufficiently small fixed sampling time.
From the above equation we have the following discrete time linear time

varying dynamic system for the sampled concentration data:

$$\begin{aligned}
&\rho_n(t+1) \\
= \; &[1 + T\Big(2a(C_d cos\gamma + Rcos(\omega Tt + \frac{2\pi}{N}n) - x_s) \\
&+b(C_d sin\gamma + Rsin(\omega Tt + \frac{2\pi}{N}n) - y_s)\Big)|V_n| sin(\omega Tt + \frac{2\pi}{N}n) \\
&-T\Big(b(C_d cos\gamma + Rcos(\omega Tt + \frac{2\pi}{N}n) - x_s) \\
&+2c(C_d sin\gamma + Rsin(\omega Tt + \frac{2\pi}{N}n) - y_s)\Big)|V_n| cos(\omega Tt + \frac{2\pi}{N}n)]\rho_n(t) \qquad (14)
\end{aligned}$$

That is, the discrete time system that can be used in the ASV to estimate the sampled concentration data is of the following form:

$$\begin{aligned}
\rho_n(t+1) &= f_n(t, a, b, c, x_s, y_s, \omega, V_n, R, C_d)\rho_n(t) \\
z_n(t) = \rho_n(t), & \qquad (15)
\end{aligned}$$

where $z_n(t)$ is the observation signal, the function $f_n(t, a, b, c, x_s, y_s, \omega, V_n, R, C_d)$ is a nonlinear continuous function of its arguments (and is defined from (14)). Note that the parameters ω, V_n, R, C_d are known, where the parameters a, b, c are not known but are subject to slow change, and the pair (x_s, y_s) is not known but is fixed. It is also reasonable to assume that $a_{min} \le a \le a_{max}$, $b_{min} \le b \le b_{max}$, $c_{min} \le c \le c_{max}$, $x_{min} \le x_s \le x_{max}$, and $y_{min} \le y \le y_{max}$ with known a_{min}, a_{max}, b_{min}, b_{max}, c_{min}, c_{max}, x_{min}, x_{max}, y_{min}, and y_{max}. Note that since the function f_n is a nonlinear function it is possible, assuming similar assumption, to replace to above described source flow model by different ones.

Coding techniques

In this section, we present some of available techniques for real time reliable communication of dynamical system over packet erasure channels. These techniques are presented for the following cases: 1) $\beta = 0$ which corresponds to the case that does not use feedback channel at all, and 2) $\beta = 1$ which corresponds to the case that uses (noiseless) feedback channel all the time.

Coding scheme - $\beta = 0$

For this case we use the coding scheme of [7], as described below:
Without loss of generality and for simplicity of presentation, consider the

following discrete time system

$$\begin{cases} x(t+1) = Ax(t), \quad x(t) = \begin{pmatrix} x_1(t) \\ \vdots \\ x_N(t) \end{pmatrix} \\ y_i(t) = [Cx(t)]_i, \ i = 1, 2, ..., N, \end{cases}$$

with $x_i(0) \in [0,1]$. For a system with $x_i(0)$, which is not in the set $[0,1]$, we use the following transformation: $r(t) = \Phi(x(t) - E)$, where the matrix Φ is invertible and Φ and E are chosen such that $r_i(0) \in [0,1]$. Note that $[V]_i$ denotes the ith component of vector V. $x_i(0)$ has the following binary representation:

$$x_i(0) = \sum_{j=1}^{\infty} w_{ij} 2^{-j}, \quad w_{ij} \in \{0,1\}.$$

The codeword $\delta_i(t)$ with length $\mathcal{R}_{it} = \lfloor R_i(t+1) \rfloor$, $0 < R_i \leq 1$ is obtained from the following linear operation:

$$\begin{pmatrix} \delta_i(0) \\ \delta_i(1) \\ \vdots \\ \delta_i(t) \\ \vdots \end{pmatrix} = M_i \bigoplus \begin{pmatrix} w_{i1} \\ w_{i2} \\ \vdots \\ w_{il} \\ \vdots \end{pmatrix}, \text{ with } M_i = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\ \vdots & & & & & & & & \\ 1 & \dots & 1 & 0 & 0 & \dots & 0 & 0 & 0 \\ \vdots & & & & & & & & \end{pmatrix},$$

where 0s and 1s on the lower triangular side of the matrix M_i are generated randomly; but the transmitter and the receiver know the components of this matrix a priori. Note that in the above equation the operator $\bigoplus$ acts as follows:

$$\delta_i(0) = w_{i1}, \ \delta_i(1) = (0 \ w_{i2}), \ ..., \ \delta_i(k) = (w_{i1}...w_{il}), ...$$

At the time instant t, the decoder can use all the received codewords up to time t, i.e., $\bar{\delta}_i(0)$, $\bar{\delta}_i(1)$, ..., $\bar{\delta}_i(t)$ to reconstruct $\hat{x}_i(0|t)$; and subsequently outputs:

$$\hat{x}(t|t) = A^t \hat{x}(0|t), \quad \hat{x}(0|t) = \begin{pmatrix} \hat{x}_1(0|t) \\ \vdots \\ \hat{x}_N(0|t) \end{pmatrix}.$$

The decoder ignores the packets containing erased bits; and produces $\hat{x}(0|t)$ using the packet received correctly. To understand how the decoding operation works, let us assume that for the ith component, the codeword $\delta_i(1)$

contains erased bits. Then, the decoder ignores $\bar{\delta}_i(1)$ and uses the following linear system of equations to reconstruct $\hat{x}_i(0|t)$.

$$\begin{pmatrix} z_i(0) \\ z_i(2) \\ \vdots \\ z_i(t) \end{pmatrix} = \bar{M}_i \begin{pmatrix} w_{i1} \\ w_{i2} \\ \vdots \\ w_{il} \end{pmatrix},$$

where $z_i(t)$ is the decimal representation of the binary string $\delta_i(t)$, and $\bar{M}_i$ is the matrix M_i without the second row (which corresponds to $\delta_i(1)$) with $\frac{1}{2^p}$ corresponding to each 1 located at the pth column of the matrix A. That is,

$$\bar{M}_i = \begin{pmatrix} \frac{1}{2} & 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\ \vdots \\ \frac{1}{2} & \dots & \frac{1}{2^p} & 0 & 0 & \dots & 0 & 0 & 0 \\ \vdots \end{pmatrix}.$$

Consequently, using the above equation, the decoder estimates w_{i1}, w_{i2}, ..., w_{il} as follows:

$$\begin{pmatrix} \hat{w}_{i1} \\ \hat{w}_{i2} \\ \vdots \\ \hat{w}_{il} \end{pmatrix} = (\bar{M}_i^{tr} \bar{M}_i)^{-1} \bar{M}_i^{tr} \begin{pmatrix} z_i(0) \\ z_i(2) \\ \vdots \\ z_i(t) \end{pmatrix},$$

and subsequently, it outputs $\hat{x}_i(0|t) = \sum_{j=1}^{l} \hat{w}_{ij} 2^{-j}$.
As shown in [7] for this coding scheme, we have

$$E||x(0) - \hat{x}(0|t)||^2 \le c^2 t 2^{-2\beta'(R,N)t},$$

where $c > 0$ is a constant depending only on $R = \sum_{i=1}^{N} R_i$ and α (erasure probability); and

$$\beta'(R, N) = \min\{\frac{R}{N}, \frac{1}{2} \min_{0 \le \eta \le 1} H(\eta||1-\alpha) + [\eta - R]_+\},$$

where $H(x||y) = x \log_2 \frac{x}{y} + (1-x) \log_2 \frac{1-x}{1-y}$ and $[x]_+ = \max\{0, x\}$.
Now, we have the following proposition for mean square reliable communication. Note that from the Chevishof inequality [4], it follows that mean square reliable communication implies almost sure reliable communication, i.e.,

$$||x(t) - \hat{x}(t|t)|| \to 0, \text{ as } t \to \infty, \text{ almost surely.}$$

Proposition: Consider above coding scheme. Using this coding scheme and by proper selection of R; and therefore, $\beta'(R,N)$, we have mean square reliable communication; and therefore, almost sure reliable communication.
Proof: We have the following inequality:

$$\begin{aligned} E||x(t)-\hat{x}(t|t)||^2 &\leq ||A||^{2t}c^2 2^{-\beta'(R,N)t} \\ &= (\sigma_{max}(A))^{2t}c^2 2^{-\beta'(R,N)t} \\ &= c^2 2^{(2\log\sigma_{max}(A)-\beta'(R,N))t}, \end{aligned} \tag{16}$$

where $\sigma_{max}(A)$ is the biggest singular value of the matrix A. Now, if $\beta'(R,N) > 2\log\sigma_{max}(A)$, then the right hand side of (16) converges to zero, as $t \to \infty$. That is, we have mean square reliable communication.

Remark We have the following remarks regarding above coding scheme:
1) Among available coding schemes, which do not use feedback channel and provide real time reliable communication for dynamical systems, above coding scheme provides the fastest convergence rate.
2) From [7], it follows that the number of binary operations required to compute the channel input $\delta_i(t)$ follows from a binomial distribution with parameter $\mathcal{R}_{it}$ and $\frac{1}{2}$. On the other hand, to do decoding operation at time t, we can use Gaussian elimination and back substitution, which requires at most $o_i t^3$; and at least $q_i t^2$ binary operations $(o_i, q_i > 0)$. Nowadays, processors use cash memory and energy management techniques. Therefore, the number of binary operations does not represent the coding computational complexity. Therefore, the coding computational complexity must be determined by simulation.

Coding scheme - $\beta = 1$: For this case we use the coding scheme of [26], as described below:
Without loss of generality and for simplicity of presentation, consider the following discrete time system

$$\begin{cases} x(t+1) = Ax(t), & x_i(0) \in [-L_i(0), L_i(0)] \\ y_i(t) = x_i(t), & i = 1, 2, ..., N, \end{cases}$$

with the matrix A diagonal ($A = diag(A_1, ..., A_N)$).
At the time instant $t = 0$, the set $[-L_i(0), l_i(0)]$ is partitioned into $2^{\mathcal{R}_i}$ equal size, non-overlapping subintervals and the center of each subinterval is chosen as the index of that interval. Upon observing $y_i(0)$, the index of the subinterval, which includes $y_i(0)$, is represented by $\mathcal{R}_i$ bits and transmitted

via the channel. If erasure does not occur, the decoder can identify the index of the subinterval, where $y_i(0)$ is located and the value of this index is chosen as $\hat{x}_i(0|0)$. Consequently, the decoding error for this case is bounded above by

$$|x_i(0) - \hat{x}_i(0|0)) \leq \frac{L_i(0)}{2^{\mathcal{R}_i}}.$$

If erasure occurs, then $\hat{x}_i(0|0) = 0$; and therefore, $|x_i(0) - \hat{x}_i(0|0)| \leq L_i(0)$. Hence, we may write $|x_i(0) - \hat{x}_i(0|0)| \leq V_i(0)$, where $V_i(0) = L_i(0)M_i(0)$ and $M_i(0)$ is a R.V. satisfying $M_i(0) = 1$ if erasure occurs, and $M_i(0) = \frac{1}{2^{\mathcal{R}_i}}$ if erasure does not occur.
When the receiver receives a packet correctly (without erasure), it sends back an acknowledgment bit to indicate the packet was received successfully. Consequently, at the time instant $t = 1$, the transmitter can compute $\hat{x}_i(0|0)$. For the time instant $t = 1$, we have:

$$|y_i(1) - A_i\hat{x}_i(0|0)| \leq |A_i|L_i(0) = L_i(1).$$

Therefore, encoder partitions the interval $[-L_i(1), L_i(1)]$ into $2^{\mathcal{R}_i}$ subintervals. Upon observing $y_i(1)$, the index of the subinterval, which includes $y_i(t) - A_i\hat{x}_i(0|0)$ is encoded into $\mathcal{R}_i$ bits and transmitted. If erasure does not occurs, $\hat{x}_i(1|1)$ will be the index of the subinterval, which contains $y_i(1) - A_i\hat{x}_i(0|0)$, plus $A_i\hat{x}_i(0|0)$. But, if erasure occurs, then $\hat{x}_i(1|1) = A_i\hat{x}_i(0|0)$. Consequently, the decoding error is bounded above by:

$$|x_i(1) - \hat{x}_i(1|1)| \leq V_i(1) = L_i(1)M_i(1),$$

where $M_i(1) = 1$ if erasure occurs, and $M_i(1) = \frac{1}{2^{\mathcal{R}_i}}$ if erasure does not occur. Following the procedure, as described above, we construct the following sequence $\{\hat{x}_i(1|1), \hat{x}_i(2|2), ...\}$.

Proposition: Using above coding scheme, we have almost sure reliable communication; and therefore, mean square reliable communication if $(1-\alpha)\mathcal{R}_i > \log|A_i|$, $\forall i \in \{1, 2, ..., N\}$.
Proof: It follows from ([26], Proposition 4.2).

Remark: We have the following remarks regarding above coding scheme:
1) This scheme is optimal in the sense that reliable communication is achieved by transmission with the minimum required bits.
2) This coding scheme is recursive; and therefore, it has low computational complexity.

Appendix II

Targets model

Model of the cars

The model of choice used to approximate the behavior of the cars is the unicycle model. It is a fifth order model having as states the world position $x(t), y(t)$, the world orientation $\Phi(t)$ and the translational and rotational velocities $v(t)$ and $\omega(t)$.

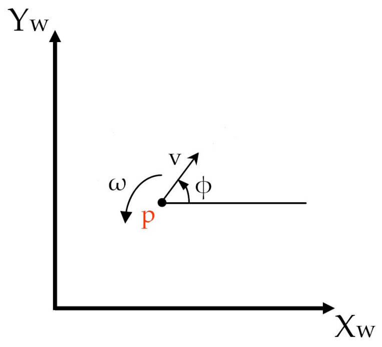

Figure 5.2: State variable definitions of the unicycle model

The continuous time nominal model is given as a set of nonlinear differential equations

$$\dot{\mathbf{x}}(t) = \begin{bmatrix} \dot{x}(t) \\ \dot{y}(t) \\ \dot{\Phi}(t) \\ \dot{v}(t) \\ \dot{\omega}(t) \end{bmatrix} = \begin{bmatrix} v(t) \cdot \ cos(\Phi(t)) \\ v(t) \cdot \sin(\Phi(t)) \\ \omega(t) \\ 0 \\ 0 \end{bmatrix} = \mathbf{f}(\mathbf{x}(t)). \tag{17}$$

The nominal model (17) does not take into account that the real car has inputs and model mismatches. In order to consider them after a discretization step we add process noise $\mathbf{w}_k$ which is composed as follows

$$\mathbf{w}_k = \begin{bmatrix} w_{x,k} \\ w_{y,k} \\ w_{\Phi,k} \\ w_{v,k} \\ w_{\omega,k} \end{bmatrix} \tag{18}$$

where $w_{x,k}$, $w_{y,k}$ and $w_{\Phi,k}$ are the scalar noise components acting on the position and orientation that represent the model mismatch whereas $w_{v,k}$

and $w_{\omega,k}$ model the accelerations acting on the car. We assume the noise sequence $\{\mathbf{w}_0, \mathbf{w}_1, ...\}$ to be uncorrelated in time. Two different approximations of the time derivative have been investigated, namely the Euler forward approximation (19) and the trapezoid approximation (20)

$$\frac{\mathbf{x}_{k+1} - \mathbf{x}_k}{T_s} = \mathbf{f}(\mathbf{x}_k) + \mathbf{w}_k \tag{19}$$

$$\frac{\mathbf{x}_{k+1} - \mathbf{x}_k}{T_s} = \frac{\mathbf{f}(\mathbf{x}_{k+1}) + \mathbf{f}(\mathbf{x}_k)}{2} + \mathbf{w}_k \tag{20}$$

where T_s denotes the sampling time.

Model(1).A

The resulting discrete time system of the first approximation is straightforward to derive since it is explicit, namely

$$\begin{array}{rcl} x_{k+1} & = & x_k + T_s v_k \cdot \cos(\Phi_k) + T_s\, w_{x,k} \\ y_{k+1} & = & y_k + T_s v_k \cdot \sin(\Phi_k) + T_s\, w_{y,k} \\ \Phi_{k+1} & = & \Phi_k + T_s\, \omega_k + T_s\, w_{\Phi,k} \\ v_{k+1} & = & v_k + T_s\, w_{v,k} \\ \omega_{k+1} & = & \omega_k + T_s\, w_{\omega,k} \end{array} \tag{21}$$

Model(1).B

Applying the trapezoid method is a bit more cumbersome since it is an implicit approximation ($\mathbf{x}_{k+1}$ appears on both sides of the equation). In a first step one obtains

$$x_{k+1} = x_k + T_s \frac{\cos(\Phi_{k+1})v_{k+1} + \cos(\Phi_k)v_k}{2} + T_s\, w_{x,k} \tag{22}$$

$$y_{k+1} = y_k + T_s \frac{\sin(\Phi_{k+1})v_{k+1} + \sin(\Phi_k)v_k}{2} + T_s\, w_{y,k} \tag{23}$$

$$\Phi_{k+1} = \Phi_k + T_s \frac{\omega_{k+1} + \omega_k}{2} + T_s\, w_{\Phi,k} \tag{24}$$

$$v_{k+1} = v_k + T_s\, w_{v,k} \tag{25}$$

$$\omega_{k+1} = \omega_k + T_s\, w_{\omega,k} \tag{26}$$

Plugging now expression (26) into (24), we get

$$\Phi_{k+1} = \Phi_k + T_s \left(\omega_k + \frac{T_s}{2} w_{\omega,k}\right) + T_s\, w_{\Phi,k} \tag{27}$$

Now all the terms are available to eliminate the $k+1$ terms on the right hand side of (22) and (23). After plugging in and rearranging we get the final set of equations as

$$\begin{aligned}
x_{k+1} &= x_k + \\
&T_s\left(\frac{1}{2}\left[(v_k + T_s w_{v,k})\cos\left(\Phi_k + T_s(\omega_k + w_{\Phi,k}) + \frac{T_s^2}{2} w_{\omega,k}\right) + v_k\cos(\Phi_k)\right] + w_{x,k}\right) \\
y_{k+1} &= y_k + \\
&T_s\left(\frac{1}{2}\left[(v_k + T_s w_{v,k})\sin\left(\Phi_k + T_s(\omega_k + w_{\Phi,k}) + \frac{T_s^2}{2} w_{\omega,k}\right) + v_k\sin(\Phi_k)\right] + w_{y,k}\right) \\
\Phi_{k+1} &= \Phi_k + T_s(\omega_k + w_{\Phi,k}) + \frac{T_s^2}{2} w_{\omega,k} \\
v_{k+1} &= v_k + T_s\, w_{v,k} \\
\omega_{k+1} &= \omega_k + T_s\, w_{\omega,k}.
\end{aligned} \tag{28}$$

Note that the accelerations at time k clearly influence position and orientation at time $k+1$. Due to the double integrator structure of the model (17), the Euler discretization has the drawback of not considering this influence already at time $k+1$ when computing the position and orientation. The more precise trapezoid approximation does not have this problem which is why we used (28) for our tracking algorithms.

The measurement model is the following. Each camera directly measures the first three states of the target, i.e. the world position and orientation. If we assume the measurement to be corrupted by additive noise $\mathbf{v}_{i,k}$, we can write for the measurement $\mathbf{z}_{i,k}$ of camera i

$$\mathbf{z}_{i,k} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix} \mathbf{x}_k + \mathbf{v}_{i,k}. \tag{29}$$

Model of the robots

Model(2)

The dynamics for the center of each robot $i \in \{1, ..., n\}$ over the time is modeled by the stochastic difference equation with sampling time T_s

$$\begin{bmatrix} x^{(i)}_{e,k+1} \\ y^{(i)}_{e,k+1} \\ \phi^{(i)}_{e,k+1} \end{bmatrix} = \begin{bmatrix} x^{(i)}_{e,k} + \bar{v}^{(i)}_{e,k}\sin(\phi^{(i)}_{e,k})T_s \\ y^{(i)}_{e,k} + \bar{v}^{(i)}_{e,k}\cos(\phi^{(i)}_{e,k})T_s \\ \phi^{(i)}_{e,k} + \Omega^{(i)}_{e,k}T_s \end{bmatrix} \tag{30}$$

where, the evader centers $[x_{e,k}^{(i)}, y_{e,k}^{(i)}]^T \in X_G$, with X_G the white ground plane set, the orientation $\phi_{e,k}^{(i)} \in [-\pi, \pi]$, the evader linear velocity $\bar{v}_{e,k}^{(i)} \in V \subseteq \mathbb{R}$, with V the admissible velocity set for the robots. The evader angular velocity $\Omega_{e,k}^{(i)} \in W \subseteq \mathbb{R}$ is i.i.d according to $\mathcal{N}(0, w_{\phi}^{(i)})$. This represents process noise associated with movement of the evaders.