Self-* Properties of Multi Sensing Entities in Smart Environments

by

Arnaud Pilpré

Diplôme d'ingénieur in Computer Science, 2001
ESIEA – Ecole Supérieure d'Informatique, Electronique, Automatique

Submitted to the Program in Media Arts and Sciences,
School of Architecture and Planning,
on May 6, 2005 in partial fulfillment of the requirements for the degree of
Master of Science in Media Arts and Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2005

Author: _____
Arnaud Pilpré
Program in Media Arts and Science
MIT Media Laboratory

Certified by: _____
V. Michael Bove, Jr.
Principal Research Scientist
MIT Media Laboratory

Accepted by: _____
Andrew Lippman
Chairperson
Departmental Committee on Graduate Students
MIT Program in Media Arts and Sciences

Self-* Properties of Multi Sensing Entities in Smart Environments

by

Arnaud Pilpré

Diplôme d'ingénieur in Computer Science, 2001
ESIEA – Ecole Supérieure d'Informatique, Electronique, Automatique

Submitted to the Program in Media Arts and Sciences,
School of Architecture and Planning,
on May 6, 2005 in partial fulfillment of the requirements for the degree of
Master of Science in Media Arts and Science

Abstract

Computers and sensors are more and more often embedded into everyday objects, woven into garments, "painted" on architecture or deployed directly into the environment. They monitor the environment, process the information and extract knowledge that their designed and programmers hope will be interesting. As the number and variety of these sensors and their connections increase, so does the complexity of the networks in which they operate. Deployment, management, and repair become difficult to perform manually. It is, then, particularly appealing to design a software architecture that can achieve the necessary organizational structures without requiring human intervention.

Focusing on image sensing and machine vision techniques, we propose to investigate how small, unspecialized, low-processing sensing entities can self-organize to create a scalable, fault tolerant, decentralized, and easily reconfigurable system for smart environments and how these entities self-adapt to optimize their contribution in the presence of constraints inherent to sensor networks.

Thesis Supervisor: V. Michael Bove, Jr.
Principal Research Scientist
MIT Media Laboratory

Self-* Properties of Multi Sensing Entities in Smart Environments

by

Arnaud Pilpré

The following people served as readers for this thesis

Advisor: _____

V. Michael Bove, Jr.
Principal Research Scientist
MIT Media Laboratory

Reader: _____

Alex (Sandy) Pentland
Professor of Media Arts and Sciences
MIT Media Laboratory

Reader: _____

Robert Laddaga
Research Scientist
MIT CSAIL

ACKNOWLEDGMENTS

Through the course of my studies, I had the chance to interact with many people who have influenced me greatly. Without their guidance, help and patience, I would never have been able to accomplish the work of this thesis. One of the pleasures of finishing is this opportunity to thank them.

To Mike Bove, my advisor, for his guidance and encouragement during my graduate studies.

To Robert Laddaga and Sandy Pentland, for doing me the honor of acting as readers of this thesis.

To my colleagues and friends, thanks for the fun and support. I want to thank especially the Object-based Media group, Ben, Diane, Erica, Gauri, Jacky, Jim, Jim and Tracy, as well as Josue, Matt, Sam and Steven, who have contributed to the SAS project.

To the students, faculty and staff of the Media Lab in general, for making this place unique. To Pat Solakoff and Linda Peterson, for their patience, their kindness and their ability to answer my many requests.

To Pr. Ojika, Dr. Jean-Christophe Terrillon, Michael Naimark and Marie Sester, for their advice and encouragements.

To my family and especially to my parents, without whom none of this would have even been possible. To my mother, for giving us all her time and attention. To my father, for sharing his passion for always knowing more.

I must give immense thanks to my wife Chiyomi and our daughter Erika, for their love, their patience and their unwavering support.

The work in this thesis was supported by the CELab, Digital Life, and Things That Think consortia, and by a joint project with the Information and Communications University and a NASA subcontract through Old Dominion University.

To the memory of my grandfather Alexandre Roger

CONTENTS

IMAGES AND FIGURES

1 INTRODUCTION

"Millions of human hands at work,
billion of minds... a vast network,
screaming with life: an organism.
A natural organism."

Max Cohen, in Pi

Smart rooms are areas furnished with sensors connected to computers that try to sense people's actions in the space and to react intelligently to them. Current architectures are usually built around powerful PC workstations that capture and process several high-level sensor inputs simultaneously. PC stations have to be installed in an adjacent room, which makes the system difficult to deploy. Worse, the failure of one of these stations would make the whole system non operational. Finally, the system usually requires human presence in all phases of operation, from deployment to maintenance.

With colleagues from the Object-Based Media Group of the MIT Media Laboratory, I have developed a set of sensing entities, called tiles, for the Smart Architectural Surfaces (SAS) project. Each tile is built around sensors (camera, microphone, sonar and thermistor), a PDA-based processing unit, effectors (speaker and screen) and a wireless networking device. These entities collaborate to create a modular system that aims to make intelligent rooms smarter.

Such intelligent sensor-based systems must be deployed in an unorganized way to eliminate the need for careful planning, decrease the cost of installation and ensure high-reconfigurability. While operating, sensing entities have to deal with hard-to-model perturbations in the environment, unpredictable changes of user priorities, sudden shifts

in resource availability, and unexpected internal errors. They need to react to these events as a whole, in real-time. Moreover, extremely complex interactions and interrelations that arise between the subsystems prohibit human intervention. All these issues urgently require the use of algorithms that favor the emergence of self-* properties[1] in complex systems such as the SAS.

I propose to investigate how small, unspecialized, low-processing sensing entities can self-organize to create a scalable, fault tolerant, decentralized, and easily reconfigurable system for smart environments [Bove04] and how these entities self-adapt to optimize their contribution in the presence of constraints inherent to sensor networks. Within the range of available sensing capacities, we intend to focus primarily on how tiles perceive their environment through image sensors as video streams provide rich and complex information about the changing environment. More particularly, we investigate how entities collaborate to achieve a common, coherent goal and how these subsystems self-adapt to deal with changes in the environment. We also describe some real-world applications that can benefit from this new approach.

Our work is at the junction of diverse but strongly interconnected domains, each one being itself the subject of advanced research and abundant literature. The second chapter gives a brief overview of these domains. We start by describing Smart Rooms, their objectives and briefly discuss some recent implementations. We then go on to show how technological developments helped the emergence of Sensor Networks and give a list of challenges that need to be solved before turning promises into reality.

The third chapter introduces our work on the SAS system, and shows how it differs from other sensing/processing/effecting architectures for smart environments. We give a complete description of the modular elements we built, as well as the multi-agent based software architecture. We conclude this chapter by highlighting what automation, through self-* properties, could bring to distributed sensing networks.

[1] pronounced "self-star", it captures many notions such as self-managing, self-tuning, self-adapting, self-organizing, self configuring, self-healing, ... into a single meta-concept

The fourth chapter describes a coarse-to-fine video-based self-localization algorithm that runs in a distributed fashion and makes the calibration of the SAS completely automated. We analyze the benefits and limitations of the current implementation and propose some improvements to make the self-localization procedure more accurate and more robust.

The fifth chapter presents a self-adaptive face tracking system. A mean-shift based tracker follows the most prominent face. The color model on which it is based is periodically updated to cope with environmental changes. In addition, two face detectors run in parallel to reinitialize the tracker if its performance decreases.

Finally, the sixth and last chapter details ongoing and future work. Some closing remarks are also included in this chapter.

Whenever possible, we try to evaluate the performance of the system and to provide evidence for the merits of distributing the processing among small sensing entities, of using self-adapting machine vision algorithms, and combining inputs from different sensors.

2 BACKGROUND

"A human being should be able to change a diaper, plan an invasion, butcher a hog, conn a ship, design a building, write a sonnet, balance accounts, build a wall, set a bone, comfort the dying, take orders, give orders, cooperate, act alone, solve equations, analyze a new problem, pitch manure, program a computer, cook a tasty meal, fight efficiently, die gallantly. Specialization is for insects."

Robert A. Heinlein

2.1 Smart Rooms

2.1.1 What is a Smart Rooms

Smart rooms are defined as environments equipped with sensors, computers and effectors that monitor the activity of people and respond to it. Their first goal is to assist people. Applications range from automating basic components such as heating or air-conditioning, to improving people's efficiency at work, by, for example, memorizing and archiving meetings, to monitoring individuals' health.

To achieve these goals, smart rooms need a way to sense people's activity. Human-centered systems use cameras, microphones and other sensors to capture and analyze people's motion, their gestures, their facial expressions, the way they talk and so on. Environment-centered systems focus instead on modifications to the "negative space": how furniture and other objects in the environment are manipulated, and try to induce the inhabitant's actions.

2.1.2 The Building Blocks of Intelligent Rooms

2.1.2.1 Sensors as the Eyes and Ears of Attentive Systems

Sensors are the gates from the real environment to the digital world. They can collect data on temperature, humidity, radiation, lighting, fields, forces (pressure, vibrations, wear fatigue...), motion, orientation, distance to another entity and so on [Fraden2004]. With recent advances in electronics, most sensors are small enough to be embedded in materials or attached to objects.

Smart Rooms are populated with sensors of different types that monitor the evolving environment and the people within it. These sensors have traditionally been connected to a network of desktop PCs that sometimes have to reside in a dedicated room near the sensing space.

With decreasing size, cost and power consumption of sensing elements, processors, memory and effectors, and the simultaneous increase in bandwidth of wireless networking technology, the deployment of hundreds or thousands of these sensors in the environment has been made possible. Interconnected through communication channels, they can form vast sensor networks that present numerous computational challenges and require that new software architectures be created to adequately process their inputs.

2.1.2.2 Processing Units for Analyzing Inputs and Generating Output

Raw information captured from the sensors is usually translated or manipulated into a required form before being transmitted to the user. The amount of computation depends on the nature of the readings and on the nature of the transformation. Readings can be as simple as a temperature value or as complex as a streaming video. Similarly, transformations can range from none (as is the case in passing a frame from the camera buffer directly to the screen buffer) to highly involved manipulations that require intensive computation.

A processing unit can read information from one or more of the sensors attached to it. Sensors can all be identical or have different types: the processing unit has to decide how to combine these different inputs and also how to generate outputs in response to these stimuli.

2.1.2.3 Effectors for Communicative Feedback

A system that can sense its environment should be able to influence it through devices called effectors (also called actuators), which are used to produce a desired change in the world, usually in response to an input. Effectors can be as limited as a few LEDs that give a simple indication of a node's internal state. Other devices that are commonly used include video screens, motor driven parts, speakers, lights and so on. Effectors are used to convey information, by, for example, displaying pictures on an LCD screen or emitting a sound, or performing some action in the environment.

2.1.3 Principal tasks performed by the Smart Environment

Some of the tasks that are performed in most Smart Room applications include:

- perceiving and identifying objects in the environment,
- detecting people: their position in the room and their proximity to objects or other people,
- identifying these people (through badges or through biometrics),
- recognizing gestures and actions: where people point, what they look at and so on,
- understanding the nature of the activity and taking appropriate measures: simply switching on a light when there is motion in a room during the night, or making a complex judgment such as inferring emotions from discussions between people.

Designing a system that can perform the tasks described above is challenging. Performance of the underlying system can usually be improved by taking contextual information into account.

2.1.4 Importance of context

By definition, "context" is the circumstances under which a device is being used, e.g. the current occupation of the user [Wikipedia]. Context-aware interfaces aim to use information related to people, who they are, where they are, what they did and what they are currently doing to improve interactions with the system [Selker2000]. This information about the context aims to simplify the reasoning about the scene and people's behavior by making assumptions about the user's current situation.

Location cues, for instance, are of particular importance in context-aware computing [Koile2003]. If the system perceives that somebody is lying still on a bed, it can infer that the person is probably resting or sleeping. If the person is lying on the floor of the bathroom, he has more likely fainted.

Understanding context should be done automatically, so as not to disturb the user by asking him to explicitly state personal information, such as his location or his mood. The process of collecting information that contributes to the understanding of context is usually done through a network of sensor nodes.

2.1.5 Some Smart Environment Projects

A numbers of *smart rooms* ([Maes1994], [Brooks1997], [Kidd1999]), *intelligent office* systems have been developed since the "Media Room" project ([Bolt1984]) of the MIT Architecture Machine Group. Following the seminal work of Pierre Wellner on The Digital Desk [Wellner1993] that aimed to merge the physical and the virtual world in an office environment, there have been many developments and advances in the field. Smart rooms have since developed into specialized areas such as smart office, intelligent kitchen [Bonanni2005] and aware meeting rooms [Moore2002] and have grown to fill entire homes or vehicles.

Smart room projects can be roughly classified related to the complexity of their sensing elements. Mozer's work [Mozer2005] shows how an intelligent house can adapt to its

inhabitants and control basic elements such as heating and light based without using any specific sensors. The system reasons entirely from the use of manual appliance controls. The Sensing Room [Mori2004] tries to understand people's activity through their direct interactions with the space. Pressure sensors on the floor can monitor inhabitants' exact location in the room. Micro switch sensors are attached to appliances and can communicate their state to the central system. Every object is tagged and its location can be inferred when sensed by the tag-reader. Similarly, the House_n project at MIT [MunguiaTapia2004] senses changes in people's "activities of daily living" using a network of sensors attached to doors or objects. Aware Home Research Initiative [Kidd1999], at the Georgia Institute of Technology, is a project directed toward assisting aging people in their everyday activities. An entire house is filled with tracking capabilities based on cameras, ultrasonic sensors and smart floor. Microsoft's EasyLiving project [Krumm2001] is a smart environment that demonstrates many sensing modalities inside a living room. Most of the sensing is done by cameras and microphones. The researchers also introduce a special interface, the XWand that a user can manipulate to control devices in the room. The system captures the readings from orientation and position sensors placed on the XWand and tries to determine the user's intention. More recently, the Agent-based Intelligent Reactive Environments (AIRE) project at MIT has focused on the development of agent-based pervasive computing systems to deal with information from various multi-modal sensors.

2.1.6 Problems with Current Smart Rooms

2.1.6.1 Issues with sensing modalities

Recent advances in wireless communication devices, sensors, hardware (MEMS) technology, made it possible to envision large scale, dense, ad-hoc networks acting as the high resolution eyes and ears of the surrounding physical space, turning the vision of Mark Weiser [Weiser1991] into reality. These sensor networks that populate Smart Rooms can usually be separated into two families.

At one extreme, some smart rooms are populated with elements that have very limited sensing capabilities. Each element must be carefully attached to a given location, as a living environment usually prohibits a high-density deployment of nodes that would have the ability to self-localize through local communication. These sensors can acquire low-level information such as motion or the presence or absence of light and may infer some basic contextual information through their "hard-coded" locations. Observations are indirect and make it difficult to recognize complex activities. The processing units, usually attached to the sensors, are generally limited and not suitable for a thorough analysis of their high-level inputs sensors.

On the other side of the spectrum, some sensor networks used in Smart Rooms are made of expensive sensing devices such as high-resolution cameras that capture rich information about the environment. By networking and calibrating several image sensors, multi-camera based computer vision algorithms can generate stereo and depth information and robustly track people in space and time ([Karuppiah2001], [Krumm2000], [Stillman1998]). Cameras are usually connected to powerful workstations that can acquire one or more video streams synchronously and process them in real-time. The processing elements are generally costly and bulky, and sometimes necessitate their own separate room near the sensing space, making the architecture difficult to scale and reconfigure.

In our opinion, there should be a trade-off between the "small and dumb" category and the "costly and bulky" one.

2.1.6.2 Issues with effectors

People who design smart rooms have usually focused more on incorporating many types of sensing modalities than on using the effecting modules. Though there are certainly situations where the system must be as transparent as possible to users, we believe the importance of effectors has been underestimated.

Furthermore, when effectors are incorporated into a system, their configuration is generally not optimal. If the position or orientation of a video camera and of a screen that displays frames from this camera are too different, the user may have to deal with a delicate mental mapping of what he sees and how he perceives himself.

Another issue with most of the Smart Rooms implementations is the strong decorrelation between sensing and effecting modalities. The two types of elements have been strongly decoupled, both logically and physically, making interaction with the system somewhat cumbersome.

2.1.6.3 Enabling the Architecture to Evolve

While fields such as material sciences and information technologies have been evolving faster than ever, creating a tremendous number of new sensing and processing devices that are continuously being made smaller, smarter and cheaper, the design of habitations for which these elements are produced has remained strangely static. A few architects, such as Neil Spiller, envision new kinds of architecture, where buildings would adapt and evolve on their own, like organisms. Sensors, processors and actuators would be integrated parts of the structure, enabling it to react to people's activity, be transformed by people and to modify itself.

2.1.6.4 Dealing with real-life environments

Computer technologies are usually built on the assumptions that the set of possible inputs is limited and that there is a well-defined response for each of these inputs. This works well for constrained applications such as word-processing, accountancy or 3D modelling software, where the user is always given the choice between a limited set of options. Applications that deal with sensing real-life environments, such as those that run in Smart Rooms systems, introduce some challenging problems, as inputs are usually noisy and events are impossible to predict or model accurately. Most of the systems installed in smart rooms are brittle: quick changes of illumination may completely disrupt a video-based tracker for instance.

2.2 Sensor Nodes

2.2.1 A Sensing/Processing/Effecting/Communicating device

Sensor networks are connected, through their sensors and actuators, to the physical space. They create a path between environmental phenomena and one or several observers to whom they report and they can form autonomous wireless sensor networks where entities collaborate with each other to collect, process and disseminate environmental information [Bharathidasan2003]. The current trend is to build nodes from small, low-cost, failure-prone sensors attached to a low-processing unit with memory, some communication devices and sometimes effectors. Nodes in the network can be identical or they can be different (have different sensors, effectors, processing devices such as in the bYOB project [Nanda2004]). In the latter case, there may be a need for a higher level of organizing resources.

2.2.2 Benefits of a distributed architecture over a centralized architecture

What are the benefits of using a distributed architecture rather than a centralized one? In general, a centralized architecture is simpler to implement but if the central node fails, the whole system ceases to work. A distributed architecture may be more expensive to design and more delicate to manage, but is likely to make the system more scalable and robust [Pottie2000]. In a distributed architecture, each sensor node captures a very localized measurement. As the number of sensor nodes increases the system can provide greater accuracy and finer resolution in the information gathered compared to what a single sensor would obtain, and can cover large areas over long periods of time. Moreover, new sensor nodes can be added dynamically, and the system continues to provide some information even if some nodes stop working or if the performance of the network is affected. In addition, if a sensor node of a dense sensor network fails, a nearby node could substitute for it and keep capturing the local information that is likely to be similar to what the faulty node would have sensed. Redundancy at this level is desirable, as it makes the whole system more fault-tolerant.

2.2.3 Sensor Network Projects

With all the exciting promises of sensor networks, there have been numerous recen projects in the field, each with its own sensing capabilities and architecture, developed with different concerns and objectives in mind. The projects can be classified by different criteria such as the density of the nodes, the energy requirements, the environment that they are designed for, the way they operate and the architecture supporting the process. We give below a few examples of sensor network projects, arranged by increasing order of the size of nodes and of the complexity of their components.

2.2.3.1 Paintable computing

Butera envisions sand grain-sized sensing nodes that can be bought in bulk and painted on surfaces to make them smart [Butera2002]. Each node has a microprocessor, limited memory, a wireless transceiver for local communication and a way to harvest power from its environment. Butera described a programming model employing a self-organizing ecology of these miniaturize nodes that supports a variety of applications.

2.2.3.2 Smart dust

The Smart Dust project aims to enclose sensors and a communication device based on lasers into a cubic millimeter package. These nodes could then be attached to food packages for monitoring product quality or glued on fingernails to sense finger motion. Several of these nodes can create a smart mesh that could make a room aware of people's presence.

2.2.3.3 Pushpin computing

Lifton introduced Pushpin Computing, where each element can be made of reconfigurable modules [Lifton2002]. The processing capabilities of each element being limited, inputs and outputs are minimal. Sensing is done through a light sensor module and feedback is given through 5 LEDs. Elements can exchange information with other

elements in their neighborhood through infrared. The pushpins have been used as a first testbed for the Paintable Computing project. Though each node is identical, they can be specialized by receiving particular sensing information or by uploading new code.

2.2.3.4 Robomote

The Robomote project at USC [Dantu2004] aims to show that the ability of a sensor node to move in its environment is critical in sensor networks. A Robomote is based on a Berkeley mote connected to a mobile base. The mote can be programmed to control the wheels of the base and move in any direction.

Other projects are closely connected to Robomote. We can cite the efforts of CotsBots at Berkeley [Bergbreiter2003], Micabot at Notre Dame [McMickell2003], Millibots at CMU [Bererton2000].

2.2.3.5 Eyes Society

Mallett developed a set of mobile robots attached to a rail that coordinate themselves to track entities in a room [Mallett2003]. Each node has a pan-tilt camera that is used as the primary sensing modality. Computer vision applications being computationally intensive, the processing is, at least partially, done on a PDA computer attached to the robotic camera node. Collaborating with other sensing elements, they can form a hybrid ecosystem that senses the environment.

2.2.4 Applications of Sensor Networks

Sensor networks' three main functions are to monitor space, to monitor things in that space (human beings, objects...) and finally to monitor the interactions of things with each other and the surrounding space. Monitoring can identify changes and report unusual events among other functions.

2.2.4.1 Habitat or environmental monitoring

The main use of sensor networks is to monitor the environment. They can control traffic [Knaian1999], send alerts in case of earthquakes, gather data in inhospitable locations, capture micro-climate data over national parks, or serve as the eyes and ears of intelligent homes [Mainwaring2002]. Some predict that, with the help of miniaturization, sensors will soon be spread on buildings to monitor deformations of their structure [Butera2002]. Space agencies are also showing interest in sending mobile sensor nodes to distant planets to lower the risks inherent to a fatal malfunction of a unique, centralized system.

2.2.4.2 Supply chain management

By affixing passive identification tags onto products, it is possible to simplify the supply chain management in a warehouse. Sensor networks help to keep updates of the inventory, and ease the order and delivery of products. Accelerometers could be fixed to expensive products and monitor the way they are handled and record details about possible mishandling of the product, such as location and time [Gaynor2004].

2.2.5 Challenges

The benefits that arise from the development of such distributed sensor network will, of course, have associated costs, and building a system comprised of many limited entities does present a number of challenges. These challenges are discussed in the sections that follow, and, in some cases, approached to mitigating the challenges are suggested.

2.2.5.1 Network deployment

Sensor networks can be made of thousands or more nodes that sometimes monitor difficult to reach spaces where there is generally no adequate infrastructure. Deploying these nodes in the space should be largely automated without human supervision to decrease the cost of installation and the need of careful planning.

2.2.5.2 Scalability

As distributed wireless network can incorporate thousands of sensor nodes or more, sensors can be clustered into groups to achieve scalability [Iyengar2003]. Each cluster can focus on specific areas or events and share only a small fraction of their information with other networks. This hierarchization of subsystems could help to provide information at different resolutions, depending on users' wishes and available resources.

2.2.5.3 Adaptivity

The sensor network should adapt to events that occur in the system (node failure, phenomenon or observer moving, decrease in bandwidth, dynamic task assignment, introduction or removal of nodes). It should monitor its constituent parts and tune workflow to achieve predetermined system goals, to optimize its performance and offer reliable and predictable performance even if each component in the system is individually unreliable. It can, for instance, modify its communication requirements in response to degrading network conditions or balance its workload to nodes that are closer to a moving tracked object.

2.2.5.4 Optimization

The system should constantly try to optimize its resources in order to minimize power, cost and network traffic loads, while ensuring reliable communication and services. This can be done by evaluating the data and transmitting only information that is relevant to the current application.

2.2.5.5 Latency

Sensor readings may be relevant only within a given time period, so there is a need to have a good measure of the latency of the system.

2.2.5.6 Uncertainty

Sensors have a limited precision and resolution and each reading is approximated and should be given with an error margin. Readings from sensors may also be corrupted: for example, an ultrasonic sensor may be facing the ground or a temperature sensor might be placed just below an air conditioning system.

2.2.5.7 Limited Computational Resources

As sensor nodes have limited processing, most of the applications cannot run on one node and should be distributed among the participants. In addition, as their memory is restricted, nodes cannot stock all the data they sense. Collaboration between entities is therefore vital. Each node should be given a sense of its relationship with others and the motivation to evaluate and maximize its contribution to the group.

2.2.5.8 Programming

There is currently a lot of research on how to efficiently program a Sensor Network. Most of the current approaches are limited to the problem they try to solve. There is a strong need to find general algorithms that can engender coherent behavior in massive sets of faulty components [Boulis2002], while hiding the complexities of the system from the developer [Newton2005].

2.2.5.9 Synchronization

Some applications are built on the assumption that nodes are synchronized. Information fusion is usually performed on sensor readings that were taken at the same time. Maintaining synchronization in a massive network of unreliable nodes is hard.

2.2.5.10 Fusion

Data fusion consists of letting independent sensors make a local decision and then combining these decisions at a fusion center to generate a global decision. A sensor

network should answer questions no single device could answer, in an organic way: each additional contributor potentially adds detail, increases precision or brings technical capacity to the overall system. There are many challenges in developing new distributed data fusion techniques to merge information from thousands of sensors, especially in managing redundancy: with many sensor nodes dispersed in a limited area, information from the different entities may be highly similar. The system should be able to recognize the redundancy and limit the transfer of low-entropy information between nodes.

2.2.5.11 Query

The user perceives the system as a whole, and the system should answer the user's queries unambiguously. Data delivery to the observer can be continuous, event-driven, observer-initiated or hybrid. The reaction to a query can vary from a simple return of a sensor value to a complex unfolding of a distributed algorithm that promotes collaborative processing among other sensor nodes. Whereas in traditional database systems queries are optimized in order to minimize computation time and disk access, queries in sensor networks must be optimized to limit power consumption.

2.2.5.12 Automation

One of the most fundamental challenges is that a sensor network should work with minimal human interaction. Configuration, as we saw before (and dynamic reconfiguration due to internal or external changes), maintenance and operation should be autonomous as the size and number of sensors usually preclude human intervention.

2.2.5.13 Robustness, security and privacy

Security is a critical factor in sensor networks, as sensors may be deployed in hostile areas where communication is monitored and nodes may be captured by the enemy. Internodal communication should be cryptic, without significantly increasing the data exchanged between nodes. A captured node should disable itself, and the network should be aware of the loss of this node.

2.2.5.14 Network longevity and cost

Sensor networks should work while being energy and communication efficient, fault-tolerant, and transparent, with limited intrusion in the environment, at a limited overall cost. System lifetime should be on the order of weeks or months, requiring low-power hardware, power-aware software solutions and, when applicable, power foraging capabilities. To save energy, a node may stay off most of the time and be switched on only for a fraction of a second. It may also limit its communications to the strict minimum. Its actions are usually constrained and very localized. Finally, the cost of the system needs to stay low, even if the number of nodes in the system becomes large.

3 THE SMART ARCHITECTURAL SURFACES PROJECT

"We live in a very fluid and dynamic world.
We live in a world that's about hot 'desking.
You come in and plug in your laptop,
you work for a bit, you move on to somewhere else.
We need to create dynamic spaces that allow those interactions to occur."

Neil Spiller, Architect

3.1 Goal

The Smart Architectural Surfaces (SAS) project hopes to fill a gap in the family of sensor networks for Intelligent Spaces. We propose a new hardware and software node architecture that makes the whole system physically and logically reconfigurable, highly automatized, more resistant to faults and to internal and external changes.

Distributed communication and control of sensor networks will benefit immediately and substantially from ongoing research in various subfields, including distributed control, agent-based simulation and hierarchical programming.

3.2 Hardware architecture

3.2.1 Anatomy of a tile

The SAS is based on units called tiles. Each tile is built around sensors (camera, microphone, sonar, thermistor and humidity sensor), a PDA-based processing unit, effectors (speaker and screen) and a wireless networking device (Figure 1). Tiles are

unspecialized. Each tile has the same devices and runs the same code, and is thus interchangeable with any other tile. The SAS differs from other sensor networks discussed earlier. The size of each node is comparable to that of a tablet PC, which is relatively large compared to nodes in other similar experiments.

3.2.2 Characteristics of the system

From a physical point of view, the SAS can be seen as:

- a network of dense sensor nodes (a microphone and a camera array),
- a wall of effectors (a giant screen or an array of speakers),
- a cluster of PDAs,
- a testbed for distributed sensing applications.

Such arrays of sensors and actuators, adequately programmed, offer a broad range of new possibilities in fields such as e-learning, man-machine interaction and collaboration. Figure 2 shows a normal setup in an office room.

3.2.3 Benefits

In addition to the benefits offered by a sensor network architecture that were described in section 2.2.2, the physical design of the SAS itself brings some additional improvements over existing sensor network projects.

3.2.3.1 Reconfigurability

Tiles can be easily snapped in and out of a grid. The tile's packaging and limited cabling make the system easy to deploy and to reconfigure. The cost of a module is low, making it affordable to build a system comprised of tens of nodes. Each node can dynamically form logical clusters with other nodes that share the same interest.

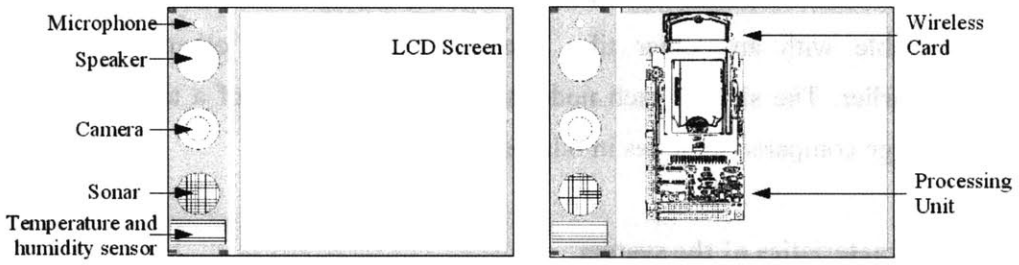


Figure 1: Anatomy of a tile - Outside and Inside view

3.2.3.2 Improving the role of effectors

Whereas most of the existing smart rooms focus on sensing capabilities, effectors play an important role in the SAS. Speakers and LCD screens may be actively used to help process information [Dalton2005], to increase the degree of autonomy of the system, and to give feedback to users. With effectors spread among sensors, the walls of the room become a truly reactive surface.

3.2.3.3 Robustness

By giving each node enough processing power to handle the processing of high-level inputs, we ensure that it can provide a minimum service by itself, though inter-tile collaboration is required when real-time performance is expected. Regularly stacked on the wall, the nodes form a dense mesh of identical sensing devices. The redundancy obtained helps to make the system more robust, though it also requires a careful optimization of resources.

3.2.4 Challenges

The SAS faces most of the challenges detailed in section 2.2.3, though the nodes are currently electrically connected to external power, which removes constraints related to energy consumption. In addition to the problems previously discussed, the architecture of the nodes and the nature of the applications we propose to run on the SAS have spawned new challenges.

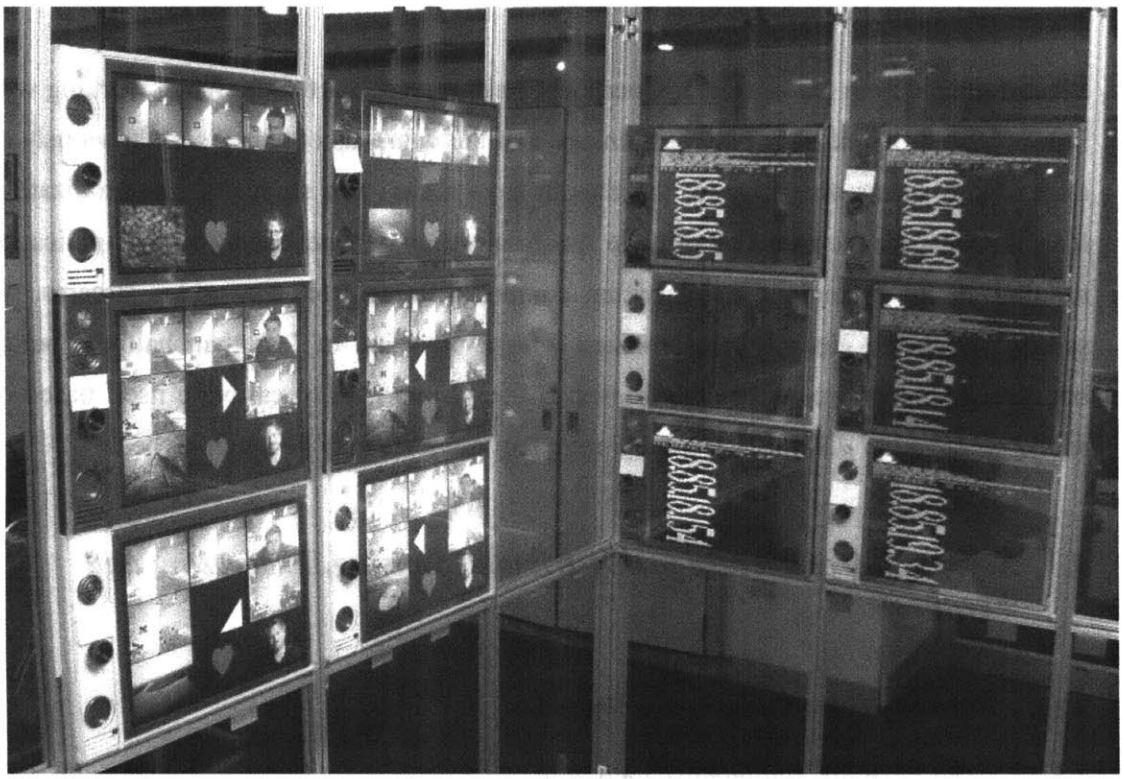


Figure 2: The Smart Architectural Surfaces system in action

3.2.4.1 Limited performance of nodes

A tile, by itself, can perform only minimal tasks and needs to collaborate with others to solve problems. Its perception of the environment is rather limited, due partly to the narrow field of view of its camera and the low sensitivity of its microphone. Moreover, the PDA that forms the processing unit of each node is not powerful enough to run intensive computer-vision/audio based applications in real-time.

The limited bandwidth imposes restrictions on how much data tiles can exchange. It is, for instance, undesirable to stream raw video from one tile to another. Tiles must pre-process raw data from the sensors and exchange only information that is relevant to the current application.

3.2.4.2 Synchronization

Measurements from the sensors are pertinent only at a certain time *t*. When working with inputs from different tiles, one must ensure that the data was obtained simultaneously on all nodes. It seems necessary for the tiles to have access to a source of synchronization. Requirements for the precision of node synchronization depend largely on the application running on the system; if an algorithm computes stereo from two video streams, the system should be globally synchronized with a precision of less than half the period between grabbing two successive frames. If the system tries to calibrate itself based on sound propagation [Dalton2005], the timing requirements may be much higher.

3.3 Software Architecture for Distributed Sensing, Processing and Feedback

3.3.1 Towards Fully Autonomous Systems

3.3.1.1 The Push for Autonomic Computing

Sensor networks are composed of a large number of diverse subsystems. Extremely complex interactions and interrelationships that arise between the subsystems prohibit human intervention. Deployment, configuration, operation and maintenance must be largely automated.

Another major shortcoming of traditional artificial systems is their inflexibility and their passive and event-driven reaction to errors. Such systems know a limited collection of process patterns and they trigger predefined actions when they encounter these patterns. While operating, sensing entities have to deal with hard-to-model perturbations in the environment, unpredictable changes of user priorities, sudden shifts in resource availability, and unexpected internal errors. They need to react to these events as a whole in real-time, and adapt themselves in a way similar to biological systems. This is the domain of autonomic computing, a term that refers to the ability of a system to manage itself.

3.3.1.2 Self- properties*

The term *self-* properties* incorporates these notions together with connecting concepts such as self-stabilization, self-configuration, self-tuning, self-adaptation, self-organization, self-healing etc. into a single meta-concept. Here, we will provide more details on the most common properties, how they relate to each other and their importance in sensor network systems.

For Pottie, the goal of a sensor network is *"to determine which of several hypothesis {hi} are true given a set of observables {Xj}"* [Pottie2000]. As each device has a limited view of the system and available resources and low latency requirements prevent it from sending all the data to a central node for processing, nodes have to self-organize to exchange only relevant information.

This notion of self-organization has been much discussed but has proved difficult to study scientifically [Collier2004]. The term "self-organization" itself is used in a variety of distinctways, some more abstract than others, and refers to a series of attributes found in complex systems:

- Self-configuration and self-reconfiguration - the notions that a given system determines the arrangement of its constituent parts by itself under varying and even unpredictable conditions (this term can be used both in the logical and the physical sense [Walter2000]);

- Self-localization - the notion that randomly distributed nodes recursively determine, without external help, their position with respect to their neighbors until a global coherent positioning of all nodes is reached;

- Self-optimization - the notion that the system can automatically monitor and control resources to ensure optimal functioning with respect to the defined requirements;

- Self-tuning - the notion that a system can change its parameters by itself, in search of improved performance and efficiency;

- Self-awareness - the notion that a system has a sense of its own character or behavior, a knowledge of its components, its current status, its goal, and its connections to other systems;

- Self-maintenance - the notion that a given system actively preserves itself, its form, and its functional status over time;

- Self-healing - the notion that a system can repair itself, or at least take actions so that faults that appear in the system do not hinder the application currently running;

- Other notions are connected to the ideas of self-organizing systems, though not completely synonymous: this include self-(re-)production, self-regulation, self-steering, self-reference, self-empowerment, self-protection.

Some other concepts are related, though they are not used as synonyms:

- Self-stabilization - the notion that a system that enters some kind of illegitimate state recovers within a finite time and returns to a legitimate state without any external intervention [Dolev2000]. Some argue that this notion is orthogonal to the concept of self-organizing systems [Adept2004], which by definition should be relatively insensitive to perturbations and should be able to restore themselves.

- Self-adaptation - the notion that systems should have the ability to monitor the state of the computation, to diagnose problems and to make changes to correct deviations from the desired behavior. Such systems usually have different ways to

solve a problem and can make a decision on which method to use at a certain time. This redundancy helps to make the system more robust to unexpected events that can occur in systems such as sensor networks. Self-adaptation is often seen as an essential component of a self-organizing system.

Whitaker insists on the fact that the different concepts mentioned above are not mutually exclusive, and that any approach to treating a system as a set of self-organizing entities should, therefore, consider which of these connotations are being addressed, as well as what features of the given system are being explicitly addressed as "self-organizing" [Whitaker1995].

We agree most closely with the general definition given by Collier, who more technically qualifies a system as self-organizing *"if a collection of units coordinate with each other to form a system that adapts to achieve a goal more efficiently"* [Collier2004]. Self-organization does not happen automatically in just any population of autonomous agents, but depends on some basic principles. Again, according to Collier, there is a certain number of conditions that must be met for that definition to hold:

(a) The system must have inputs and some measurable output.

(b) The system must have a goal or goals.

(c) The units that compose the system must change internal state based on their inputs and the states of other units.

(d) No single unit or non-communicative subset of units can achieve the system's goal as well as the collection can.

(e) As it gains experience in a specified environment, the system achieves its goals more efficiently and/or accurately, on average.

3.3.1.3 Self-evaluation and peer-monitoring

Whenever possible, the agent also evaluates its performance at its current job. In some instances, several algorithms may perform the same task using different approaches, and it is possible to analyze the performance of each of them and to give more computation time to those that perform best in the current conditions. The ability of a program to evaluate itself and to change its behavior to improve its performance is of particular importance for the kind of problems we want to investigate. Each sensor entity must be aware of how well it processes data and what contribution it brings to the group. Tiles also monitor the state of their peers and may reorganize their interaction patterns if a component breaks down or if new components are added to the system.

3.3.2 A society of cooperative agents

3.3.2.1 Distributed intelligence through a multi-agent system

Sensor networks are comprised of many elements and the performance of the system depends on the ability to control the separate elements in a distributed fashion. An appropriate way to handle complex issues inherent to decentralized systems is to base the architecture on an ecology of distributed agents [Minar1999]. An agent is an automatic process that can communicate with other agents to perform some collective task. A single agent is usually limited by its knowledge and its computational resources. As the domain becomes more complex, a Multi-Agent System (MAS) is needed to address the reasoning task efficiently, in the same way that a living organism is composed of largely autonomous and diverse parts. A MAS is a suitable architecture for sensor networks, as it effectively implements ideas on automation and collaboration. In a MAS, there is no global control, the location of data is decentralized and computation is asynchronous. Agents can be homogeneous or heterogeneous and may have common or different and sometimes conflicting goals. Each agent has incomplete information or capabilities for solving the problem.

Current challenges include improving communication protocols, designing negotiation

mechanisms and establishing coordination strategies for problem solving and investigating how multiple agents can collectively reason about the state of the domain based on their local knowledge, local observation through the sensors and limited communication. There is some interesting current research on collaboration. One approach is based on simulating free market systems that imitate our own trade system: price information, linked to a given task, is spread across the network and agents try to maximize their profit, i.e. their reward for their effort, by bidding on these tasks. They can then subcontract them to other agents [Dias2002].

3.3.2.2 Inter-thread and inter-tile communication

The software architecture is based on a set of agents, each responsible for a sensing, processing or effecting task. Each agent runs in its own thread and is allocated a specific processor time slot, which can increase or decrease depending on its performance. In most of our applications, we have around 10 agents running on each tile. Their functions are as diverse as reading sensor inputs, displaying buffers on a part of the screen, sending a video-frame to another tile, participating in the self-localizing phase of the system, evaluating global performance, modifying the agents time slots and so on.

Agents can exchange information such as time-stamped video frames. If two agents are running on the same tile and need to exchange large amounts of data (such as images or big chunks of audio), shared memory is the preferred way to make the information available to both agents. Semaphores are used to ensure mutual exclusion when threads read and write into the shared memory location. Threads can also exchange messages, such as requests or queries. The message passing classes have been extended to inter-tile communications and an agent can transparently collaborate with agents on other nodes as if they were running on the same tile. Agents can also emit requests. If the bandwidth decreases, an agent can ask its peers to limit their communication, or compress images at a lower quality before sending them to another tile. It can also request other agents to be created on the same tile or on a remote node to accomplish a given task.

3.3.3 Distributed processing

3.3.3.1 Leader election

Some applications require that a tile is elected as a leader to coordinate further efforts. Leader election is a basic problem of distributed algorithms and it has been extensively studied. Some implementations of the electing process have been as simple as picking the tile with the "highest" IP address, though there has also been considerable research on designing more general approaches that could also handle temporary or permanent failure of the leader, and eventually a re-election among the remaining nodes.

3.3.3.2 Distributed consensus

Consensus refers to *"the problem of achieving coherency among nodes of a distributed computer system"* [Wikipedia]. Achieving consensus is particularly challenging when the number of nodes grows or the reliability of nodes decreases. Tiles may need to agree on the value of parameters, such as the level of image compression for all video frames transiting through the communication channel.

3.3.3.3 Synchronization

Tiles need to be synchronized, especially due to their close coupling with the physical world. This is to ensure coherence between the information captured by the sensors so that it can be compared on similar grounds. Because each application has different time synchronization requirements, it is desirable to control the level of precision we want to reach. Due to clock drift, each tile uses *ntp* to regularly synchronize its clock to a common clock in the system. Agents running on one tile may have to reach a consensus on how precise they want the synchronization to be, depending on the goal they are trying to achieve. It is more complicated than just deciding on the highest precision required by the set of agents. The cost of improving the synchronization increase exponentially with the level of precision desired, and an agent requiring a high level of synchronization but doing a poor job due to faulty sensors or bad positioning should therefore not be allowed its desired synchronization level.

3.3.3.4 Processing information in a distributed fashion

There are many issues when considering distributed implementation of algorithms in a sensor network: which sensor should sense, what needs to be sensed, what information should be shared between nodes (raw data or results), how to exchange the data when the bandwidth is limited, how to distribute hypotheses and models. In the SAS, sensing, processing and displaying are done collaboratively, with subsystems dynamically forming groups to solve given problems [Bove2004]. Tasks that require the combined effort of two or more tiles, such as any form of distributed sensing or distributed computing, need protocols and structures that provide coordination and methods of aggregating measurements from distributed sensors nodes.

Sensor fusion, also known as data fusion, is *"the process of collecting, distilling and displaying information from a number of homogeneous or heterogeneous information sources (sensors) to produce an integrated output that is more accurate and complete than that achievable using the same sensors independently of one another"* [Wikipedia].

Sensor fusion is vital as it helps to minimize the network bandwidth. There have been many strategies developed to achieve sensor fusion. Some are based on exchanging low dimensional feature vectors while others rely on exchanging likelihood values. Recent methods are based on probabilistic models and Bayesian data fusion algorithms.

3.4 Some applications

Many applications could benefit from the SAS system. The network of cameras, microphones and sonars could be used to locate and track entities in a room based on a combination of cues such as their shape, the sound they emit, and their motion. The SAS can also estimate which tile has the best view of a given face and base further processing, such as face recognition or facial feature detection, on video frames from this tile. Similarly, it can use its array of microphones to enhance somebody's speech. Screens can display large, complex scenes and speakers can render special audio effects. Sonars can

be used as low-processing alarms that would "wake-up" higher level sensors when an entity starts moving in front of them.

3.5 SAS as a Network of Cameras

Among all the sensors attached to each tile, we decided to focus on cameras: seeing the SAS as a network of "intelligent" visual sensors. The main challenge is to develop some distributed computer vision applications that can run on low-processing nodes that communicate through limited bandwidth.

Until now, array processing was mostly out of reach for consumer applications, perhaps due to the significant cost of dedicated hardware and the complexity of processing algorithms. Such an ad-hoc sensor/actuator network can be used to capture/render different audio-visual scenes in a distributed fashion leading to novel emerging applications.

3.5.1 Distributed Vision Systems

As the field of view of a camera is relatively limited, one needs to find way to improve the system's perception. Solutions include using fisheye lenses or mounting a camera onto a pan tilt support. Fisheye lenses introduce distortions that need to be corrected before processing frames, and do not provide depth information. Pan/tilt cameras can be used to get a larger view of a static scene and to track entities as they move in the environment, but, unless they are affixed to a mobile system, their perception is still limited.

Multi-camera systems are based on a network of cameras distributed around a given scene. The cameras perceive activities and aggregate information from multiple viewpoints. This helps to reduce the uncertainty about the scene: for instance, an entity that is partially or entirely invisible in the field of view of a camera may be seen by another image sensor. In most multi-camera systems, the video cameras are linked to powerful stations that can process multiple video streams simultaneously, in a centralized

way. In a sensor-network configuration, processing of multiple video streams requires some special attention to what should be processed and what should be exchanged, due to the processing and bandwidth limitations.

3.5.2 Previous work

Devious [Romig1995] was one of the first distributed systems for computer vision applications and used the idle time of a distributed network of workstations to run image-processing tasks. Based on the fact that most computer vision algorithms can be broken down into smaller tasks that can be efficiently achieved in parallel, Devious was dedicated to low-level processing of images, such as computing FFT or detecting edges. Given the limited power of workstations used at that time, Devious could not be adapted to real-time performance.

There have been several attempts at modeling vision systems as a set of multi-agents ([Boissier1997], [Yanai1998]). Graf introduced a multi-agent system architecture, each agent being responsible for a specific vision task [Graf1999]. Master agents bid on vision tasks and can assign sub-tasks to other agents. Agents can be added or deleted at run-time, making the system truly dynamic. Here again, the agents are mostly working in parallel on the processing of images captured by one camera. Knoll presents an extended framework that was applied to an intelligent assembly setup made up of a complex set of cameras, some fixed and others placed on robotic arms [Knoll2001]. Fusion of inputs from the different cameras helps improving the visual servoing and the detection of occluded objects.

4 DISTRIBUTED SELF-LOCALIZATION

"Where am I? Who am I? How did I come to be here?
What is this thing called the world?
How did I come into the world?
Why was I not consulted?
And If I am compelled to take part in it,
Where is the director? I want to see him."

Soren Kierkegaard

4.1 Self-Localization as an essential first step

4.1.1 On the importance of localizing entities in the environment

The growth of mobile technologies has driven the development of location-aware devices and applications based on these devices [Muthukrishnan2005]. Giving a device the ability to localize itself in its environment immediately creates numerous service opportunities, such as user assistance, area monitoring and contextual marketing. A prerequisite inherent in most applications running on the SAS is the ability for each tile to localize itself in relation with other tiles, and especially to determine its direct neighbors, as information captured through sensors is strongly connected to context.

4.1.2 Automating the localization process

The SAS architecture can theoretically support hundreds of tiles simultaneously across different locations, and tiles can be added or removed dynamically while the system is operating. The localization problem should, therefore, be largely automated.

There has been extensive research on self-positioning and self-localization in sensor networks and several methods have been developed. Here, we review the most common techniques.

4.2 Overview of Localization techniques

Bulusu defines localization as a mechanism to find the spatial relationship between objects [Bulusu2000]. Localization is usually seen as a two-step problem, in which we first define a coordinate system, and then calculate the distance between sensors and deduce the relative or absolute position of sensors in the coordinate system.

4.2.1 Taxonomy and System Classification

There are many ways to compare localization systems. [Muthukrishnan2005] classifies them into different categories:

- indoor / outdoor environment,
- range-based / range-free systems,
- hardware-based / software based,
- tightly coupled / loosely coupled,
- centralized / decentralized approach,
- relative / absolute coordinate system,
- coarse grained / fine grained.

The localization techniques can be compared on the basis of diverse criteria that include level of automation, accuracy, scalability, time complexity and cost.

4.2.2 Review of existing techniques

4.2.2.1 Range estimation techniques

Localization in sensor networks is most commonly accomplished using range

estimations, i.e. explicit distance calculation, between sensor nodes. Existing methods include:

- **Time of Arrival (ToA, also called Time of Flight):** Measure the signal (RF, acoustic, ultrasound) arrival time and calculate distance based on transmission times and speeds. ToA requires synchronization between the emitter (the source) and the receptor (the sensor) to accurately measure time-of-flight. Time of Flight is the technique that is used in the Global Positioning System (GPS), a satellite-based radio navigation system. A GPS receiver can determine its location on the surface of the planet by triangulating signals from satellites orbiting around earth, with a precision of 10 to 20 meters. GPS works only in outdoor environments and, due to the receiver's current size and power constraints and the high precision measurements and synchronization required by the system, is currently of limited interest for sensor networks.

- **Time Difference of Arrival (TDoA):** Measure the difference in arrival times of the same signal at two different sensors, in which case all sensors should be synchronized (single source, multiple synchronized receivers). Alternatively, measure the difference in arrival times of two signals from synchronized sources at the same sensor. In this case, all sources should be synchronized (multiple synchronized sources, single receiver).

- **Received-Signal-Strength-Indicator (RSSI):** RSSI is mainly used with RF signals. This technique is based on the change in energy of a radio signal, which is inversely proportional to distance from the signal's source. Due to multipath fading in outdoor environments, range estimation with RSSI can be inaccurate.

- **Angle-of-Arrival (AoA):** AoA allows a calculation of the angle between sensors equipped with directional antennas. The size and cost of this additional hardware may prevent it from being used in most sensor networks.

4.2.2.2 Network connectivity (range free schemes)

If a sensor cannot receive signals from enough base stations (at least 2 for AoA, at least 3 for ToA, TDoA, and RSSI), none of the previous techniques will work. In this case, network connectivity can be exploited for range estimation ([Niculescu2001]):

- **Distance Vector-hop (DV-hop):** Beacon nodes, which are nodes that have knowledge of their global position, broadcast their positions to all nodes in the network. Sensors compute the minimum distance in hops to several base stations, which compute an average distance per hop to other base stations. The base stations then send this information to the whole network allowing nodes to calculate their positions. The accuracy is directly related to the density of the network. A variant technique (DV-distance) uses cumulative range estimates from RSSI instead of hop counts.

- **Euclidean:** The Euclidean method estimates a sensor's distance to a base station based on the geometry of neighboring nodes.

4.2.2.3 Proximity

Proximity based localization simply detects an object near a known location, either by physical contact (pressure sensors, or capacitance field detectors) or contact-less scanning (RFID). The active badge [Want1992] was one of the first projects to use this principle. Each person wears a badge that regularly transmits a unique IR signal. Sensors placed at fixed positions in the environment read the signals and deduce the proximity of a person.

4.2.2.4 Scene Analysis

Localization methods based on scene analysis use environmental features to infer location. In some cases, the observed scene may be simplified in order to improve the detection of features and increase the accuracy. Sensors may be given information about their environment prior to running the localization algorithm and will then compare the information they perceived with that of the known world [Wolf2002].

4.2.3 Signals

Optical signals can be based on visible light, infrared or laser systems. This is one of the most common methods used to transmit information up to a few meters, sue to the particularly low cost of emitters and receivers. Transmission is subject to interference from other sources of light and obstacles. RF signals do not suffer from interference with obstacles, as they diffract around and pass through common building materials. Transmission of these signals is up to one hundred meters. Acoustic signals (and especially ultrasound) propagate more slowly in air than RF signals, which allows TOF to be precisely estimated from the TDOA of simultaneously emitted acoustic and radio signals. Ultrasonic ranging techniques can attain higher precision than those using audible sound, however, they provide shorter effective range and require more expensive hardware.

Inertial systems use orthogonal gyroscopes and/or accelerometers mounted on mobile elements to measure movement from a known initial position. Conceptually, acceleration is integrated to find velocity and integrated again to find position. Unfortunately, errors accumulate over time. Without recalibration, the positioning error is unbounded. Over short time periods, inertial systems can be quite accurate.

4.3 A Fast, Distributed Self-Localization Algorithm on the SAS

From among all the available localization systems, we chose to use video to self-localize the tiles in space. We propose a fast, distributed, video-based self-localization algorithm for the SAS based on coarse stereo calculation. Though using video may not be the easiest method for localizing an entity in space, it is appealing to use this modality in the SAS as each node has the required hardware to capture and process images. This is done through a two-step process: the tiles first try to find their neighbors and then go on to determine the spatial transformation with each of their neighbor. This task is primordial for many vision-based applications. For example, a multi-camera distributed application such as person tracking needs to have a precise idea on where each camera, i.e. each tile,

is located in the space so as to provide the system with a larger view of its environment and a spatial representation of the scene. This is also a critical step for grouping nodes that have a spatial relationship and for efficiently using the effectors by, for example, displaying a large image when necessary or playing a tune with spatial sound effects through the speakers.

4.3.1 Background

4.3.1.1 Video calibration

In Computer Vision, the Camera Calibration phase consists of estimating a set of parameters that describes the imaging process. These parameters can be intrinsic (aspect ratio, focal length and principal point) or extrinsic (position and orientation of the camera coordinate frame with respect to the world coordinate frame). Multi-camera calibration is a natural extension of single camera calibration, where the position of each camera is found with respect to the others. Typically, we first calculate the orientation of each camera in a common frame system and we then determine the relation between all the cameras.

Calibration is generally based on photogrammetric methods, which make use of prior knowledge about the scene and usually require additional hardware and/or precise hand measurements. Calibration may prove to be very time consuming, and may be impractical for some applications where deployment should be unplanned and human intervention minimized. Video self-calibration refers to the ability to find the camera's intrinsic and extrinsic parameters, using only the information available in the frames taken by the camera moving around a given, fixed scene.

Multi-camera self-calibration refers to the process of calculating the position and orientation of the cameras in space based solely on the images taken from the cameras, without giving the system any additional measurements. Multi-camera self-calibration may be related to research on how to infer the motion of a camera through an image sequence.

4.3.1.2 Related Work

The performance of self-calibration systems is generally poor, though it is possible to improve it by using a simple pattern with known geometry such as a checkerboard pattern ([Zhang2000], for the mono-camera calibration, which was later extended to multi-camera calibration). Zhang's method gives accurate results and is widely used, but requires some expertise and specific strategies when calibrating multiple cameras. As the checkerboard is usually seen only by a small number of cameras at each time, the calibrated structures are chained together, which reduces the accumulation of errors. Svoboda proposes a quasi-autonomous calibration system where 3D points are generated by waving a laser pointer through the volume seen by the cameras [Svoboda2005]. His technique shows sub-pixel accuracy though the calibration phase is time consuming (around 30 minutes to calibrate 16 cameras). Physical relocation of a few cameras may require the calibration algorithm to be run again for the whole system. Gandhi devised a calibration procedure for a reconfigurable array of cameras, but this method is based on omni-directional cameras [Gandhi2004]. Finally, most of the self-calibration systems are based on a centralized architecture, and there has been very little published on how to solve vision-based self-localization of a set of nodes in a distributed fashion. Choi described a distributed calibration system coined the Distributed Alternating Localization-Triangulation, where processing occurs at each node to limit communication [Choi2004].

4.3.2 From Coarse Self-Localization to Fine Self-Calibration

For high-level video-based applications such as 3D reconstruction or vision-based robotic manipulation, the results of the calibration must be as precise as possible. In other applications, we may just need to roughly position a camera with respect to another one ("Camera A is somewhere to the left of Camera B") or a slightly more precise localization ("Camera A is to the left of Camera B, but Camera C lies between them").

We are proposing an iterative, fully automated method that estimates the neighbors of each tile assuming their camera axes are somewhat parallel. We also show how this method can be extended to manage other camera configurations (in-plane and out-of-

plane rotation, as well as translation) using effectors, and to take motion into account to improve the accuracy of the calibration, though at the cost of extra-computation.

4.3.3 Overview of the distributed self-localization algorithm on the SAS

Any element of the SAS can request that members of the system self-localize. The self-localization process first tries to estimate the position of direct neighbors of each tile, relying partly on the regular arrangement of the nodes in the room. Each tile is first assigned a randomly selected peer. Both tiles in the pair synchronize themselves so that they each capture a video frame simultaneously. After compressing their respective frame, they exchange it over the network. Each element in the pair then tries to calculate the disparity between its video frame and the one sent by its selected peer. They update their knowledge based on this new information and their confidence. Both tiles are then assigned a new peer and perform the same process again. The system progressively tends to equilibrium as its subsystems start to agree on their relative position in the space. The algorithm runs in the background, exchanging frames with its neighbors to continuously monitor its state. Each tile may send a request to restart the whole self-localization procedure if results become too incoherent, following a physical reconfiguration of the nodes for instance.

4.3.4 Feature Detection

4.3.4.1 Choice of features

To evaluate the position of a camera relative to another camera, we need to estimate the displacement of features that are seen by the two image sensors. The quality of this estimate depends largely on the choice of features and the performance of the algorithm used to detect them in the image. Commonly used features include edges, corners, line or curve segments and regions.

Regions are costly to extract and may have irregular shapes, making the matching difficult. Segments, and in particular their end points, may be difficult to detect exactly.

We choose to extract corners from video frames, though they are, like edges, more likely to suffer from occlusions.

4.3.4.2 Corner detection

Several methods have been developed to detect corners in an image [Zheng1999]. They can be roughly separated into two categories: template-based corner detection and geometry based corner detection. Each technique can be evaluated with respect to quality of detection in terms of its sensitivity and precision, repeatability of results, and speed of detection. Vincent compared both the SUSAN algorithm [Smith1995], that uses brightness comparison, and the Harris method [Harris1988], based on the Plessey operator [Vincent2002]. The latter proved to be the most stable, which is a key property when the corners are used for stereo matching, and we used this method for detecting features. A standard post-processing procedure is used to ensure that corners are far away from each other.

4.3.5 Feature Matching

For each iteration, a tile detects features in the captured frame using the Harris method, as described above. It then tries to match each of these features to a point in the video frame sent by its assigned partner. This process, called Features Matching, has been extensively studied. It is usually done in two steps. Starting from the assumption that a feature in one frame can be matched with any point (or not at all) in the other frame, we generate, for each feature in the first image, some correspondence hypotheses, based on the epipolar constraint. We then reduce the search space by using additional constraints such as similarity (both features should look alike), uniqueness (usually each feature in a frame can match no more than one feature in the other frame), continuity (disparity maps are continuous almost everywhere), ordering (preservation of ordering across images), relaxation and so on. Our algorithm does not actually match features in both frames. After detecting features in the frame captured by its camera, the tile tries to track these features in the frame it received from its peers. The main step in the tracking phase is the computation of the optical flow of each feature. Though optical flow is a method that is

used for estimating visually the motion of objects by evaluating the displacement of each image pixel in an image sequence, it can be seamlessly transposed to the case of moving cameras around a static scene. Kanade proposed a method for tracking pixels on consecutive frames [Kanade1981]. It assumes that the motion of features is very limited between two frames, which in our case would correspond to assuming that the two cameras are very close to each other and pointing in approximately the same direction. Bouguet introduced a coarse-to-fine implementation of the Lucas-Kanade algorithm, using a pyramidal representation of images [Bouguet2000]. This makes the tracking more robust even if frames are widely separated. We derived a fixed-point version of his algorithm based on this pyramidal version of Lucas-Kanade algorithm. Results of the feature matching algorithm are shown on Figure 3.

4.3.6 Position evaluation

After calculating the optical flow of each feature, it is possible to roughly estimate the position of a tile related to its peer. For each feature that has been matched, we also compute the displacement vector between the two frames, which gives us a rough knowledge on the position of the tile related to its peer. As it searches for its neighbors, each tile maintains a list of the different results it gets along with each candidate's IP. After all the iterations have been completed, we can combine the partial results to deduce the position of all the tiles in the space. Each tile votes for the best candidates for each of the eight surrounding positions. Each vote can be weighed by several factors, such as the measure of confidence calculated during the feature matching phase, so as to reduce the influence of low-quality matching. The average displacement between pairs of features can also be taken into account to choose between two candidates that compete for the same position. As the number of iterations increases, each tile refines its position among the peers that surround it.

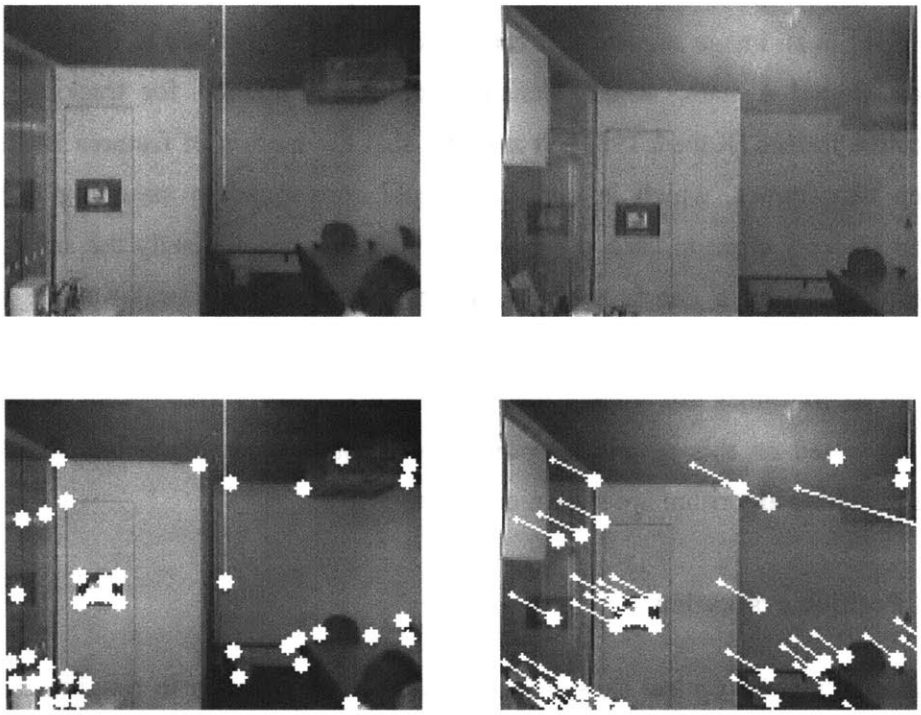


Figure 3: (From top to bottom and left to right): a pair of frames F1 and F2, results for feature detection on F1 and feature tracking on F2.

4.3.7 Peer assignment

In the first few iterations of the algorithms, pairs of tiles are assigned randomly. If each tile has to perform the localization process with all other tiles before it can evaluate its position, the self-localization process would be time consuming (linear with the number of tiles). As tiles start to have a rough idea of their position in the grid, they can request a test of their assumptions with a particular tile. By letting the tiles regularly exchange information about their localization, such as their estimated position relative to another tile, assignments can be made more intelligently, and the process can be directed to speed the stabilization of the algorithm. For example, if tiles A and C have estimated their position with respect to one another and have both estimated that A was to the left of C, C can send A its beliefs about its top and bottom neighbors, which are good candidates for A's top right and bottom right neighbors.

Tiles are proactive and always try to maximize their contribution to the system. As the

algorithm runs, the pairs of localized tiles grow into clusters of neighbored tiles until the system becomes stable (each tile knowing its neighbors). The self-localization process terminates when each tile has reached a level of confidence that is higher than a given threshold, or if all tile assignments have been tried.

4.4 Evaluation, results and future work

4.4.1 A few steps

We ran the system on a group of six tiles, as shown in Figure 4. The camera of Tile 3 was set up with an out-of-plane rotation, so as to show the robustness of the system to small variations. Figure 4 shows results for two of the five iterations. In round 1, all the pairs of tiles are able to correctly self-localize. On the other hand, in round 3, pairs (2,4) and (3,5) are unable to correctly localize themselves. Results for the five rounds of partial calibration between pairs of tiles are summarized in Table 1. Results show that individual self-localization is correct in 17 cases and almost correct (i.e. arrow pointing in almost the right direction) in 8 cases, while being wrong in only 5 cases. Combining these individual results through group voting can make the system even more robust.

4.4.2 Performance evaluation

The self-localization technique we propose is a decentralized task, the processing being distributed among the tiles. It is somewhat scalable, though exchanging video frames between pairs of tiles wirelessly currently limits the level of scalability that can be reached. The process is fully automated, except for the internal calibration of each camera that can be calculated once, with the resulting calibration matrix saved for further use. One iteration of the algorithm takes approximately 50 seconds when dealing with 40 features. The duration of the entire procedure depends on the number of tiles and on the way the pairs of tiles are formed. At each iteration, the position evaluation is done twice in parallel, but as the frame order is inverted for both tiles (each tile extracts features from the video frame it captured, then tracks them on the frame it received from the peer it was assigned to), the results can be different, making the positioning results more robust.

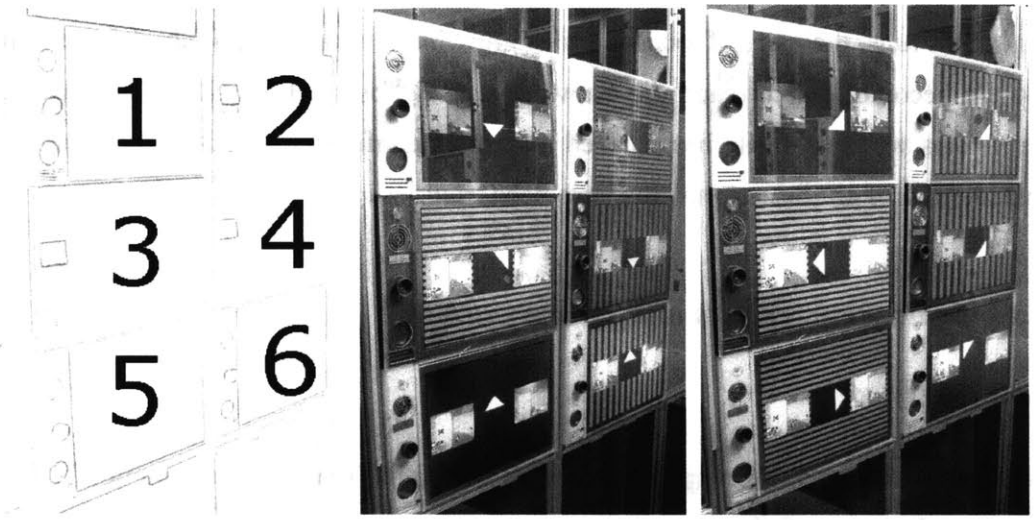


Figure 4: The setup and the results of Round 1 with pairs (1,5), (2,3) and (4,6), and Round 3 with pairs (1,6), (2,4) and (3,5). Both tiles in a pair display the same background.

4.4.2.1 Inter-tile information exchanges

At each iteration, all the tiles should transmit their frame to their assigned peer. Though we can reduce the amount of information sent by compressing the video frame, further minimization of the communication between two peers is of interest. One way to achieve this would be to have each tile detect features on the frame it captured, and then send only the features together with the set of pixels from a well-defined area surrounding each of them. Each tile would then be able to match features using cross-correlation.

4.4.2.2 Time evaluation

For the setup shown in Figure 4, each iteration, during which the algorithm finds 40 features in the first frame and tracks them into the second frame, took approximatively 50 seconds on a 400 Mhz Xscale processor. The self-localization for the set of six tiles took less than 4 minutes. Based on experiments, it appears we can lower the number of features to 20 while keeping the same success rate, though having more features makes the calibration phase more precise.

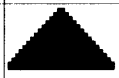
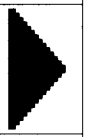
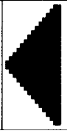
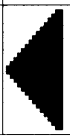
	TILE 1	TILE 2	TILE 3	TILE 4	TILE 5	TILE 6
TILE 1						
TILE 2						
TILE 3						
TILE 4						
TILE 5						
TILE 6						

Table 1: Partial calibration for the setup presented in Figure 4. This can be read as "[ColumnElement] is on the [DirectionOfArrow] of [RowElement]".

It is difficult to evaluate an upper bound for the total time needed to calibrate the system, as it depends largely on the scene. For each iteration and for each pair of tiles, the estimation is done twice, once by each element of the pair. As the frame order is different, localization results can also be different. Processing between peers occurs in parallel, so at each iteration, we can perform *n/2* unique position estimations. An upper bound for the algorithm gives a total of *n-1* iterations for each tile to be able to deduce its position, providing that the system is not reconfigured during this phase.

There is a direct trade-off between the time and performance of the algorithm. If the system can afford to spend more time self-localizing its components, each tile can be asked to extract more features, making the self-localization more precise.

4.4.3 Improvements

Though the external calibration is based on low-resolution images, our method can be extended to provide finer information on both position and orientation in the room coordinate system.

4.4.3.1 From Coarse Localization to Fine Calibration

Though the self-localization process described earlier is fast, autonomic and somewhat robust to errors, the results that are obtained are not accurate enough to perform high-level computer vision algorithms such as multi-view 3D modeling. We describe here how to determine the motion of a virtual camera moving between the positions of the camera of two tiles.

The transformation matrix between the cameras of the pair of tiles is specified by six parameters (three translations and three rotations). If we are able to correctly track one feature, this gives us four pieces of information, which are the coordinates of the feature F1 in the first image and those of the estimated position of the same feature in the second frame. This also introduces three new parameters, which are the 3D coordinates of the 3D point represented by F1 in both cameras. If we track n features, we obtain a system made of $4n$ *observations* and $5+3n$ parameters (as the transformation is given up to a scale factor). The system is solvable if we are able to track 5 features from one frame to the other, though in practice, more points are needed to cope with the noise from the observations.

From the pairs of features, we can calculate the fundamental matrix. The fundamental matrix **F** is a 3x3 matrix that encapsulates the intrinsic geometry and verifies, for each feature x and its corresponding feature x':

$$x'^{T}\mathbf{F}x = 0$$

Assuming that we know the intrinsic parameters for all cameras (i.e. we know the calibration matrix C which describes the mathematical relationship between 3D points in the world and where they appear in the image), we can determine the essential matrix **E** from the fundamental matrix **F**:

$$\mathbf{E}=\mathbf{C}'^{T}\mathbf{F}\mathbf{C}$$

The position of the cameras in space, i.e. the rotation matrix **R** and the translation vector **T** can then be retrieved from the essential matrix up to a projective transformation:

$$\mathbf{E}=\mathbf{R}[\mathbf{T}]_{x}$$

where $[\mathbf{T}]_{x}$ is the cross product matrix of vector **T**.

4.4.3.2 Improving the calibration using motion of entities in the room

Some self-calibration techniques are based on the study of entities in motion [Azarbayejani1995]. Jaynes shows how it is possible to calibrate a large network of cameras by matching trajectories instead of features [Jaynes2001]. It nevertheless involves intense calculations and requires that enough interesting trajectories are observed. The calibration phase thus takes a long time and may not be appropriate in systems that need to be quickly deployed.

4.4.3.3 Improving the performance using effectors

As we want the system to be as automatic as possible, we limit user's intervention to the

minimum. Multi-camera calibration usually requires that the user show a checkerboard pattern to the cameras. It usually leads to very accurate results, though it requires some expertise to correctly place the pattern. Assuming that tiles can be placed on all the walls of a room, one can simulate the "checkerboard" based procedure by displaying some geometric patterns on the screen attached to each tile. Modules that can see the patterns could use them to localize themselves with respect to the tile (or group of tiles) that are displaying these patterns.

4.4.3.4 Improving the performance by merging with other self-localization methods

As we saw before, there are many ways to perform localization in sensor networks, each method having its advantages and drawbacks. Intuitively, it seems interesting to have several localizing algorithms running in parallel and to combine the results. Dalton devised an audio-based self-localization algorithm for the SAS that lets the entities self-localize precisely [Dalton2005]. Tiles use both their speaker and microphone to evaluate their distance to their peers, though orientation in space is impossible to determine based solely on audio. A natural extension of our algorithm would be to fuse the information from both the video-based and the audio-based self-localization algorithms. The agent in charge of fusing the results should intelligently evaluate the contribution of each algorithm: it may give more weight to the audio-based localization algorithm if the environment is dark or prefer the video-based one if the environment is too noisy.

This would require to build a more robust framework to merge beliefs, measures of confidence and uncertainties on the position of tiles in the space.

4.4.3.5 Monitoring reconfiguration

Tiles keep self-localizing as a low-priority, background task. If a tile notices some incoherencies, such as new measurements that contradict its assumptions, it may request to restart that the self-localization phase be restarted. This may be done by learning a background model and monitoring changes over the short, medium and long term.

5 SELF-ADAPTIVE FACE DETECTION

"The wise adapt themselves to circumstances, as water moulds itself to the pitcher"

Chinese Proverb

5.1 Why Face Detection?

5.1.1 Detecting faces on the SAS

Yang defines the problem of face detection as *"determining whether or not there are any faces in a given image and, if present, return the image location and extent of each face"* [Yang2002]. Face detection is a key element in many computer vision applications but remains a challenging problem as faces have a very high degree of variability.

There are a number of applications that can benefit from detecting faces in the SAS. One of the problems we are interested in solving is to stream to a remote peer the best view of somebody's face taken by the array of cameras. Indeed, several image sensors can simultaneously observe a face from different angles, though all views may not be "interesting".

5.1.2 One Problem, Many Approaches

The approaches that have been suggested can be classified into different categories [Yang2002]:

- Top-down methods, which are based on rules that transcribe human knowledge, such as the relationship between facial features.
- Bottom-up methods, which aim to detect invariant features such as eyes or mouth and to infer from that the existence of a face.

- Template matching methods, which correlate a model of a face with a candidate and infer from the correlation values the presence of a face.
- Appearance-based methods, which learn a model of face from a set of training images.

There is no optimal method. It is now well accepted that the best results are obtained by combining several techniques, based on different cues and executed concurrently.

As faces can be anywhere inside a given frame, the detector usually divides the image into overlapping regions and tests for the presence of a face in each region. To speed up the detection process, the detector can also choose to reduce the space by pre-processing the image. It can then choose to focus only on image regions with color closest to human skin color, or on area where motion was detected, for instance.

5.2 Self-adaptive Face Detection

Computer Vision problems are notoriously difficult to solve, as "real world" environments are hard to model accurately. For certain tasks, and under well-constrained conditions, vision algorithms can perform adequately. But the performance will often degrade drastically when conditions change. Laddaga argues for the merits of a self-adaptive approach to perceptual applications [Laddaga2000].

We apply a self-adaptive strategy to the difficult problem of detecting faces in a video stream. Our system is based on a combination of three different techniques that run in parallel. The first method is a wavelet template-based detector, an optimized version of the Haar Face detector introduced by Viola [Viola2001]. The second technique uses skin color to segment skin pixels from background pixels, then tries to estimate the shape of prominent blobs of skin. Finally, the third method is a mean-shift face tracker that tries to track a face on successive frames from a single camera. Results from all detectors can be combined, and each detector has a methodology to evaluate its performance and to change its behavior accordingly.

5.2.1 Haar-based Face Detection

Viola developed a fast object detector system based on a technique called boosting [Viola2001]. Boosting is *"a machine learning meta-algorithm for performing supervised learning"* [Wikipedia]. It consists of combining weak classifiers (classifiers whose performance is slightly better than a random function) to form a robust classifier. Though this approach uses very simple features, the detector is fast and its detection rate is high. The framework was extended by Lienhart who used rotated Haar-like features to improve the classification [Lienhart2002]. We devised a face detector that follows the pyramidal implementation of the Haar-based face detector developed with OpenCV [OpenCV].

5.2.2 Skin-color based face detection

5.2.2.1 Color-based face detection

Detecting faces based on skin color is a two-step process. First, we need to differentiate between pixels that have skin colors and those that don't. We can then analyze the morphology of the clusters of skin-color and select those with a shape close to that of a face. Needless to say, there are many factors that can negatively affect the face detection process. Skin color tone depends on each individual and on the current illumination. Though the face is usually approximated as an ellipse-shaped surface, this really depends on the orientation of the face and of possible occlusions.

5.2.2.2 Skin color modeling

The first step is to classify pixels into one of two classes: skin or non-skin. To do that, we need to find a color space in which skin color is easy to model. Terrillon has compared 25 different color spaces for skin segmentation and tried to determine which ones were the most efficient for skin color-based image segmentation, based on the following criteria:

- compactness and shape of the skin distribution,

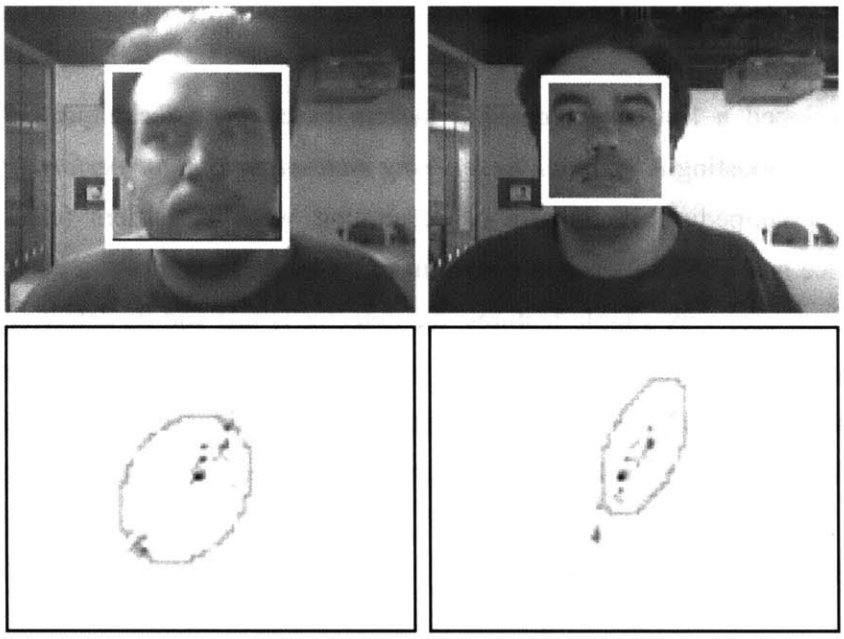


Figure 5: Haar-based Face Detection for two frames with different lighting conditions and corresponding Skin Color model

- discrimination between skin and non-skin pixels,
- robustness to change in illumination and camera systems,
- computational cost of the color space transformation [Terrillon2002].

We found that the normalized r-g, CIE-xy chrominance spaces and spaces that are constructed as suitable linear combinations or as ratios of normalized r, g and b values, offer the best trade-off and appear to be the most consistently efficient for skin color-based image segmentation. Moreover, they have the advantage of low computational cost, and hence they are particularly suitable for a system that needs to perform in real-time. We decided to use the YCbCr space to create a model for skin color. This space is perceptually uniform and separates luminance (Y) and chrominance (CbCr), i.e. lightness and color. Many studies have found that the chrominance components of skin tone color are independent of the luminance component, which allow us to take only the CbCr components into account.

We collected many skin and non-skin samples with our tiles' cameras and transformed them into the YCbCr color space. A 2D histogram representing the skin distribution in the CbCr chrominance space was computed from the samples. The distribution of skin colors is then modeled with a mixture of Gaussians using the Expectation Maximization algorithm. Each Gaussian is defined by its centroid and its covariance matrix.

5.2.2.3 Skin-color based face detection

When we try to detect faces in an image, we first convert it to the YCbCr space. Each pixel is then classified as either part of the skin-pixel class or part of the non-skin class based on the Mahalanobis distance of the pixel to each of the skin clusters found previously. We obtain a binary map that is searched for large clusters of skin pixels. Some morphological operators such as noise removal, erosion or dilation can be used to improve results. Some results are shown on Figure 6. We then label all blobs whose shape loosely matches that of an ellipse as faces.

5.2.2.4 Face verification

Classifying skin blobs as face or non-face is relatively naïve, as other body parts such as the palms of the hands, have shapes that are very similar to faces. In order to avoid this kind of confusion, the image region corresponding to the blob may be normalized and analyzed using the Haar detector. In this perspective, the color segmentation can be seen as a way of reducing the search space for faces.

One of the issues with this method is that it is difficult to automatically select "pure" skin samples. A non-negligible number of facial features or background pixels are usually included in the samples. Assuming that most of the skin samples will have a majority of skin pixels, we may disregard the clusters that are likely to correspond to the under-represented background colors.

Figure 6: Adaptive skin segmentation - left: Skin segmentation with the initial model; center: Skin segmentation after lighting conditions have changed and before the skin model is updated; right: skin segmentation after the skin model was updated.

5.2.3 Face Tracking

5.2.3.1 About Face tracking

Face tracking refers to the ability to locate a face in consecutive video frames. Whereas detecting faces in each frame implies a search over the whole frame, face tracking assumes that the motion of moving entities between two consecutive frames is limited, and that the search for a face can be conducted in a reduced region of the image. As the field of view of a camera is limited, face and people tracking algorithms have been implemented on mobile camera systems (Figure 7) and multi-camera setups.

5.2.3.2 Mean Shift Face Tracking

Our tracking module is based on the Mean-Shift (MS) algorithm [Comaniciu2000] [Bradski1998]. It is a nonparametric statistical method for seeking the nearest mode of a point sample distribution. Although the algorithm is limited to the tracking of elliptical objects, it is somewhat robust to partial occlusions and is fast to compute. We use this algorithm to track blobs of skin pixels in the frame. The ellipse that fits the detected face in the first frame is used to initialize the tracking. For each successive frame, we first convert all pixels into the YCbCr space, and then compute the binary skin map as explained in the face detection section. Classifying the skin color blobs into face or non-face is costly, and would reject faces that do not have an ellipsoidal shape. The MS algorithm first tries to cluster the skin pixels inside a search window, centered on an initial position. It then recursively moves to the kernel-smoothed centroid for every data point.

Figure 7: Face tracking with a Pan-Tilt Zoom camera

5.3 Towards a Self-Adaptive Face Tracking system

5.3.1 Self-Adapting Systems in Computer Vision

Some interesting work in the area of self-adapting systems is directly related to our research. Robertson introduces GRAVA [Robertson2004], a multi-agent architecture based on self-adaptive techniques that has been used for aerial image interpretation [Robertson2000] and face recognition [Robertson2003]. Soto proposed an adaptive visual system able to combine and exchange different algorithms according to their performance and applied it to two robotic problems: single person tracking and obstacle detection [Soto2002].

5.3.2 Characteristics of each Face Detector

The Haar-based face detector is a robust upright face detector. The original version took about twenty seconds to process a 320x240 video frame at 5 different scales on the processing unit we are using, and the current version can detect faces in an average of 0.6 seconds, which is still too slow for real-time performance. Whereas the Haar-based face detector can only detect upright faces with limited angular variations, the skin-color based detector can detect faces at any angle, as long as the skin color blob keeps its ellipsoidal shape. It is faster than the Haar-based detector, but is more sensitive to environmental changes and its performance can severely degrade if the background is complex, with entities having a color similar to that of skin. Of the three face detectors that run in

parallel, the Mean-Shift face tracker is in theory the fastest method for extracting a face in successive frames. However it needs to be properly initialized by one of the two other face detectors. It can track faces even if they are partially occluded, though the loss of the face requires the tracking to be reinitialized.

5.3.3 Self-Adaptive Multi-Camera Face Detection

In sensor networks, the increasing cost of deployment and maintenance, the strict robustness and flexibility requirements and the unpredictability of environments are important considerations. DARPA recognized the need for a new approach, and introduced self-adaptive software which *"evaluates its own behavior and changes behavior when the evaluation indicates that it is not accomplishing what the software is intended to do, or when better functionality or performance is possible"*. Some techniques for self-adaptation derive directly from machine learning or neural networks.

The performance of our system may vary because each sub-detector's performance depends directly on the environment and can degrade during the tracking, which ultimately result in losing track of the face. The Haar-based face detector, the Skin Color-based face detector and the Mean-Shift algorithm all run in parallel and can evaluate their performance at their current job and change their behavior to improve their performance.

5.3.3.1 Self-Evaluation

Each detector can evaluate its performance and measure its confidence level in the results it obtains. The Haar-based face detector relies on a measure of distance between the wavelet template and the sample it is testing. This shows how good the matching is. The skin color based face detector can keep statistics on blobs it has detected (numbers, holes and so on). Finally, the face tracking system can monitor the shape of the skin blob it is following. To perform this evaluation, a detector needs to know whether it has correctly detected a face. As we use three detectors in parallel, we can assume that a face has been correctly extracted if at least two detectors roughly agree on its position and shape. As each detector has its own time complexity, they asynchronously reach a conclusion on the

presence or absence of a face in the video frame they process. The frame on which they perform the detection is then likely to be different for each of the detectors and merging should be done carefully.

5.3.3.2 Self-Adaptation

Following its evaluation, the detector make a decision to change its behavior. The Haar-based face detector may increase the number of scales at which it detects faces to improve its performance or to decrease its computation time. The skin color based detector may decide to refine its color model to cope with illumination changes or to modify its elliptical model if necessary. Skin samples from the faces detected by the Haar-based face detector are collected and used to update the skin color model. Finally, the face tracker may increase the size of its search window if it feels the target is moving too fast, or it may require reinitialization by one of the two other face detectors.

5.4 Evaluation, Issues and Extensions

5.4.1 System performance

Figure 6 shows how the system can keep detecting faces when there are changes in lighting intensity or chromaticity. We currently have a few issues with the adaptive skin color modeling. The Haar-based face detector that serves as the first step for collecting skin samples may detect a false positive in the background (i.e. may label a non-face region as a face), and use these background pixels to update the skin color model, decreasing the quality of skin segmentation. A way to solve this problem is to decrease the number of false positives, by lowering the detection threshold, for instance. Though the tracking system works at frame rate on a desktop PC, we could not optimize it enough to reach real-time performance, while running in parallel with the other methods. The current frame rate for the tracking system on the SAS is about 2 frames per second.

5.4.2 Computation time

5.4.2.1 Optimizing the code implementation

Face detection is usually seen as an essential first step in human machine interfaces. This step should provide a fast and precise estimation of the location of a face in the video frame. As processing power on each tile is limited, it may be difficult to reach real-time performance. We had to modify all floating-point calculations into fixed-point operations. This had to be done carefully so as not to incorporate approximations that would lower the performance of the system. We also use several look-up tables to speed-up the processing, though given the limited memory space, some have to be dynamically allocated and calculated.

5.4.2.2 Devising some strategies to reduce the computation time

The Haar-based face detector is a blind search on the whole image. The approach consists of sliding a search window over scaled and rotated transformations of the original image. This is far from efficient and does not make any assumptions about physical reality. One way to reduce the computation time is to classify the image regions to be tested in decreasing order of their probability of containing a face. These probabilities may be obtained by estimating the motion of a face. Other research suggests combine notions such as saliency, gist and gaze to reduce the search space and decrease the computation time [Siagian2004].

5.4.2.3 Distributing tasks among tiles

Aside from these techniques, there are a few other ways to reduce the computation time of the face detector algorithms: we can reduce the resolution of the video frames, or, in the case of the Haar face detector, the number of scales at which the detector looks for a face. These techniques have a direct impact on the face detection rate. Another way to increase the speed of detection is to distribute the processing among a group of tiles that have some spare processing slots. All the tiles do not have the same workload at a given time: some may have not detected any motion for a long period and stay idle most of the

time while others may simply be too far from the scene to perceive anything useful. A tile could broadcast a request to other tiles and send a command attached with relevant data to the peers that are willing to help. Task allocation would be dynamic, depending on the load of each processor. The processing load may be shifted from one cluster of nodes to another as the phenomenon that is being observed moves, or as the physical configuration of nodes change.

5.4.3 Multi-Camera Face Detector

5.4.3.1 Tracking people with multiple cameras

After running the self-localization algorithm detailed in the previous chapter, tiles have a rough idea of their position in relation to their neighbors. If a face is about to leave the field of view of the camera, a tile can propagate information related to the tracked face to the neighbor (or the set of neighbors) that is likely to have the face in its field of view. Multiple cameras can also make the system more robust to occlusions as a face partly or entirely hidden in one frame may clearly appear in another.

5.4.3.2 Counting people

A direct application of face detecting and tracking systems is counting people that are present in a constrained environment. There have been several investigations of this subject [Schlögl2001]. Counting people is important in domains such as security or human-machine interaction. People-counting algorithms are usually based on a multi-camera multi-people tracking algorithm, and are more attractive than pure tracking for testing self-adaptive systems, as their outcomes are easy to evaluate. As each tile has a limited field of view, it cannot independently achieve a good estimate of the number of people in the room. It can nevertheless track faces using a multi-cue self-adaptive algorithm, which adapts to the context perceived by the system, as shown in [Spengler2000]. The system cannot, of course, simply add the number given by each tile, as some people may be perceived by more than one module at a given time. So subsystems have to collaborate and reach a consensus on their estimate of the number of people in the room.

6 CONCLUSION

6.1 Summary

We introduced the Smart Architectural System, a system made of modular elements that all have sensing/processing/effecting/communicating capabilities. They form complex networks that spawn dynamic interactions. Such systems are difficult to deploy, to manage and to maintain.

We have described how this kind of architecture can be programmed through a set of multi-agents to act autonomously, to self-organize and to self-adapt to internal and external changes. This was illustrated through two applications. First, we devised a distributed video-based self-localization system for the SAS. Tiles are randomly assigned two by two and, through a simple stereo estimation, try to estimate the 3D transformation between the two cameras, i.e. between the two tiles. The process is repeated until each tile has localized itself with respect to its neighbors. The second application attempts to implement a self-adaptive face detection algorithm. We approach the system by fusing the results from three different detectors that run in parallel and that are based on different cues. Each detector can evaluate itself and change its behavior based on its performance.

Colleagues of the Object-Based Media group and myself are still working on extending the capabilities of the SAS. We give below some of the problems we are likely to tackle in the future.

6.2 Extensions

6.2.1 Towards Self-Organized Multi-Sensor Fusion

As mentioned earlier, a microphone and sonar are also attached to each subsystem in addition to the image sensor. To improve performance, visual information can be combined and enhanced with inputs from other sensors. Combining information from

heterogeneous sensors is difficult, especially in a system such as SAS, where sensors are imprecise, uncalibrated, or faulty. In the same way that a machine vision algorithm should integrate several cues and dynamically combine them, the possibility of mixing different types of sensor output together is important and helps improve accuracy. Most data fusion techniques are based on post-perceptual integration: senses are considered distinct, each sensor input being obtained and treated independently from the others, and the final decision is made based on combination of these. New research [Coen2001] based on biological evidence shows that cross-modal integration might be better suited for multi-sensor fusion.

6.2.2 Detecting and Tracking Miscellaneous Entities

Another challenge is to extend the self-adaptive face detection framework to the detection, recognition and tracking of any entities. This requires the design of robust algorithms to identify objects from different angles of view and efficient methods for exchanging the set of features that represent an object between tiles.

6.2.3 Mobile Sensor Networks

Though the SAS can provide feedback to the user though its effectors, a module has a very limited influence on its environment. They can be passively relocated, but cannot move by themselves. There has been some recent interest in making sensing nodes mobile in the space. Mobility can overcome some of the limitations described above. It gives the system a higher degree of reconfiguration in case of failing nodes and can help improve sensing if the nodes are too far from the event they need to observe. Making the nodes able to move could nevertheless make it harder to preserve processing and connectivity. Localization, an important problem in sensor networking applications, becomes even more critical to the system. Mallett has shown how a set of mobile and static sensing entities could create an ecosystem of devices that physically and logically reorganize to improve their performance in monitoring entities and events [Mallett2004].

7 REFERENCES

[Adept2004] Rapport d'Activite du groupe Adept, IRISA Rennes, 2004.

[Azarbayejani1995] A. Azarbayejani and A. Pentland: *"Camera self-calibration from one point correspondence"*, MIT Media Laboratory, perceptual computing technical report #341, 1995.

[Bererton2000] C. Bererton, L.E. Navarro-Serment, R. Grabowski, C. J.J. Paredis and P. K. Khosla: *"Millibots: Small Distributed Robots for Surveillance and Mapping"*, Government Microcircuit Applications Conference, 20-23 March 2000.

[Bergbreiter2003] S. Bergbreiter and K.S.J. Pister: *"CotsBots: An Off-the-Shelf Platform for Distributed Robotics"*, IROS 2003, Las Vegas, NV, October 27-31, 2003.

[Bharathidasan2003] A. Bharathidasan and V. Anand Sai Ponduru: *"Sensor Networks: An Overview"*. 2003.

[Boissier1997] O. Boissier and Y. Demazeau: *"Une architecture multi-agents pour des systèmes de vision ouverts et décentralisés"*, numéro spécial sur l'Intelligence Artificielle Distribuée et des Systèmes Multi-Agents, Technique et Science Informatiques, 16(18): 1039-1062, January 1997.

[Bolt1984] R.A. Bolt: *"The Human Interface"*, Chapter 3, 35-53. California: Lifetime Learning Press, 1984.

[Bonanni2005] L. Bonanni, C.H. Lee, and T. Selker: *"Counter Intelligence: Augmented Reality Kitchen"*, Long paper in Extended Abstracts of Computer Human Interaction (CHI), Portland, 2005.

[Bouguet2000] J-Y. Bouguet: *"Pyramidal implementation of the lucas kanade feature*

tracker - description of the algorithm", Tech. Rep., Microprocessor Research Labs, Intel Corporation, 2000.

[Boulis2002] A. Boulis and M. B. Srivastava: *"A framework for efficient and programmble sensor networks"*, the 5th IEEE Conference on Open Architectures and Network Programming (OPENARCH 2002), New York, NY, June 2002.

[Bove2004] V.M. Jr Bove and J. Mallett: *"Collaborative Knowledge Building by Smart Sensors"*, BT Technology Journal, 22:4, Oct. 2004.

[Bradski1998] G. Bradski: *"Computer Vision Face Tracking For Use in a Perceptual User Interface"* - Intel Technology Journal - 2nd Quarter 1998.

[Brooks1997] R.A. Brooks with contributions from M. Coen, D. Dang, J. DeBonet, J. Kramer, T. Lozano-Perez, J. Mellor, P. Pook, C. Stauffer, L. Stein, M. Torrance and M. Wessler: *"The Intelligent Room Project"*, Proceedings of the Second International Cognitive Technology Conference (CT'97), Aizu, Japan, August 1997.

[Bulusu2000] N. Bulusu, J. Heidemann and D. Estrin: *"GPS-less Low Cost Outdoor Localization for Very Small Devices"*, IEEE Personal Communications Magazine, Vol. 7, No. 5, pp. 28-34. October, 2000.

[Butera2002] W. Butera: *"Programming a Paintable Computer"*, PhD Thesis, MIT Media Lab, Feb. 2002.

[Choi2004] H. Choi, R. Baraniuk, W. Mantzel: *"Distributed Camera Network Localization"*, Asilomar Conference on Signals, Systems, and Computers, Pacific Grove, CA, (Accepted, November 2004).

[Coen2001] M. Coen: *"Multimodal Integration - A Biological View"*, in Proceedings of IJCAI, Seattle, WA, 2001.

[Collier2004] T.C. Collier and C.E. Taylor: *"Self-organization in sensor networks"*, Journal of Parallel and Distributed Computing, 64(7):866—873, 2004.

[Comaniciu2000] D. Comaniciu, V. Ramesh, P. Meer, *"Real-Time Tracking of Non-Rigid Objects using Mean Shift"*, IEEE Conf. on Computer Vision and Pattern Recognition, Hilton Head Island, South Carolina, 2000.

[Dalton2005] B. Dalton and V.M. Bove: *"Audio-Based Self-Localization for Ubiquitous Sensor Networks"*, 118th Convention of the Audio Engineering Society, Barcelona, Spain, 28-31 May 2005.

[Dantu2004] K. Dantu, M. Rahimi, H. Shah, S. Babel, A. Dhariwal and G. S. Sukhatme: *"Robomote: Enabling Mobility in Sensor Networks"*, CRES-04-006, 2004.

[Dias2002] M. Dias and A. Stentz: *"Opportunistic optimization for market-based multirobot control"*, IEEE/RSJ International Conference on Intelligent Robots and Systems, September 2002.

[Dolev2000] S. Dolev: *"Self-stabilization"*, MIT Press, Cambridge, MA, 2000.

[Fraden2004] J. Fraden: *"Handbook of Modern Sensors: Physics, Designs, and Applications"*, 3rd ed., 2004, XVII, 589 p. 403 illus., ISBN: 0-387-00750-4.

[Gandhi2004] T. Gandhi, M. Trivedi: *"Calibration of a reconfigurable array of omnidirectional cameras using a moving person"*, Proceedings of the ACM 2nd international workshop on Video surveillance & sensor networks, October 15-15, 2004, New York, NY, USA.

[Gaynor2004] M. Gaynor, M. Welsh, S. Moulton, A. Rowan, E. LaCombe and J. Wynne: *"Integrating Wireless Sensor Networks with the Grid"*, IEEE Internet Computing, special issue on the wireless grid, July/Aug 2004.

[Graf1999] T. Graf and A. Knoll: *"A multi-agent approach to self-organizing vision systems"*, Proceedings of The Third International Conference on Computing Anticipatory Systems, Hongkong, China, 1999.

[Harris1988] C. Harris, M. Stephens: *"A Combined Corner and Edge Detector"*, Alvey Vision Conf., pp. 147-151, 1988.

[Iyengar2003] R. Iyengar and B. Sikdar: *"Scalable and Distributed GPS Free Positioning for Sensor Networks"*, Proceedings of IEEE ICC, pp. 338-342, Anchorage, AK, May, 2003.

[Jaynes2001] C. Jaynes: *"Calibration of Large Sensor Networks using Observed Motion Trajectories"*, Dept. of Computer Science, Technical Report: TR316-01, Feb 2001.

[Kanade1981] T. Kanade and B. Lucas: *"An iterative image registration technique with an application to stereo vision"*, in 7th International Joint Conference on Artifical Intelligence, 1981, pp. 674–679.

[Karuppiah2001] D. R. Karuppiah, Z. Zhu, P. Shenoy and E. M. Riseman: *"A fault-tolerant distributed vision system architecture for object tracking in a smart room"*, in Int. Workshop on Computer Vision Sys, July 2001.

[Kidd1999] C. D. Kidd, R. Orr, G. Abowd, C. Atkeson, I. Essa, B. MacIntyre, E. Mynatt, T. Starner and W. Newstetter. *"The Aware Home: A Living Laboratory for Ubiquitous Computing Research."* Proceedings of the Second International Workshop on Cooperative Buildings, CoBuild'99.

[Knaian1999] A. N. Knaian: *"A wireless sensor network for smart roadbeds and intelligent transportation systems"*, Ph.D. dissertation, Massachusetts Institute of Technology, 1999.

[Knoll2001] A. Knoll: *"Distributed contract networks of sensor agents with adaptive reconfiguration: modelling, simulation, implementation"*, Journal of the Franklin Institute, Elsevier Science, Vol. 338, No. 6, pp. 669-705, September 2001.

[Koile2003] K. Koile, K. Tollmar, D. Demirdjian, H. Shrobe, and T. Darrell: *"Activity Zones for Context-Aware Computing"*, In Ubicomp 2003 Proceedings, 2003.

[Krumm2000] J. Krumm, S. Haaris, S. Meyers, S. Brumitt., M. Hale, S. Shafer, *"Multi-Camera Multi-Person Tracking for EasyLiving"*, IEEE Workshop on Visual Surveillance, July 2000.

[Krumm2001] J. Krumm, S. Shafer and A. Wilson: *"How a Smart Environment Can Use Perception"*, UBICOMP 2001 Workshop on Perception for Ubiquitous Computing, October 2001.

[Laddaga2000] R. Laddaga: *"Active Software In Self-Adaptive Software"*, Proceedings of the First International Workshop on Self-Adaptive Software, Oxford, April 2000.

[Lienhart2002] R. Lienhart and J. Maydt: *"An extended set of Haar-like features for rapid object detection"*, IEEE ICIP, vol. 1, pp. 900-903, 2002.

[Lifton2002] J. Lifton: *"Pushpin computing: a platform for distributed sensor networks"*, Master's thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, 2002. 114

[Maes1994] P. Maes, T. Darrel, B. Blumberg, S. Pentland. "*The ALIVE system : Full-Body Interaction with Animated Autonomous Agent*". M.I.T. Media Laboratory Perceptual

Computing Technical Report No. 257, Jan. 1994.

[Mainwaring2002] A. Mainwaring, J. Polaster, R. Szewczyk, D. Culler, and J. Anderson: *"Wireless Sensor Networks for Habitat Monitoring"*, First International Workshop on Sensor Networks and Applications (in conjunction with ACM MobiCom '02). pp. 78-87, 2002.

[Mallett2003] J. Mallett and VM Bove, Jr.: *"Eye Society"* Proc. IEEE ICME 2003.

[Matsuyama2000] T. Matsuyama et al: *"Dynamic memory: Architecture for real time integration of visual perception, camera action, and network communication"*, In Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition, volume 2, Hilton Head Island, SC, June 2000.

[McElligott2002] McElligott, L., Dillon, M., Leydon, K., Richardson, B., Fernstrom, M., Paradiso, J.A.: *"ForSe FIElds' - Force Sensors for Interactive Environments"*, in G. Borriello and L.E. Holmquist (Eds.): UbiComp 2002, LNCS 2498, Springer-Verlag Berlin Heidelberg, October 2002, pp. 168-175.

[McMickell2003] M. B. McMickell and B. Goodwine: *"Micabot: A robotic platform for large-scale distributed robotics"*. In IEEE International Conference on Robotics and Automation", 2003.

[Moore2002] D. Moore: *"The IDIAP smart meeting room"*, Tech. Rep. 02-07, Dalle Molle Institute of Perceptual Artificial Intelligence (IDIAP), 2002.

[Mori2004] T. Mori, H. Noguchi, A. Takada and T. Sato: *"Sensing Room: Distributed Sensor Environment for Measurement of Human Daily Behavior"*, First International Workshop on Networked Sensing Systems (INSS2004), pp.40-43, 6 2004.

[Mozer2005] M. C. Mozer: *"Lessons from an adaptive house"*, in D. Cook & R. Das

(Eds.), Smart environments: Technologies, protocols, and applications (pp. 273-294). Hoboken, NJ: J. Wiley & Sons, 2005.

[MunguiaTapia2004] E. Munguia Tapia, S. S. Intille, and K. Larson: *"Activity recognition in the home setting using simple and ubiquitous sensors"*, in Proceedings of PERVASIVE 2004, vol. LNCS 3001, A. Ferscha and F. Mattern, Eds. Berlin Heidelberg: Springer-Verlag, 2004, pp. 158-175.

[Muthukrishnan2005] K. Muthukrishnan and M. E. Lijding and P. J. M. Havinga: *"Towards Smart Surroundings: Enabling Techniques and Technologies for Localization"*, Int. Workshop on Location- and Context-Awareness (LoCA), published by Springer-Verlag, Berlin, held in Oberpfaffenhofen, Germany, May. , 2005, pp. to appear

[Nanda2004] G. Nanda: *"bYOB (Build Your Own Bag):A computationally-enhanced modular textile system"*, Ubicomp, Nottingham, England, 2004.

[Newton2005] R. Newton, Arvind, and M. Welsh: *"Building up to Macroprogramming: An Intermediate Language for Sensor Networks"*, To appear in Proceedings of the Fourth International Conference on Information Processing in Sensor Networks (IPSN'05), April 2005.

[Niculescu2001] D. Niculescu and B. Nath: *"Ad hoc positioning system (APS)"*, in Proceedings of GLOBECOM, San Antonio, November 2001.

[OpenCV] OpenCV - Intel open source computer vision library

[Pottie2000] G. Pottie and W. Kaiser: *"Wireless Integrated Network Sensors"*, Communications of the ACM, 43 (5): 51-58, May 2000.

[Robertson2000] P. Robertson: *"An architecture for self-adaptation and its application to aerial image understanding"*, proceedings of the first international workshop on Self-

adaptive software table of contents (2000), Oxford, United Kingdom, pp 199-223.

[Robertson2003] P. Robertson and R. Laddaga: *"An Agent Architecture for Information Fusion and its Application to Robust Face Identification"*, Applied Informatics 2003: 132-139.

[Robertson2004] P. Robertson, R. Laddaga: *"The GRAVA Self-Adaptive Architecture: History; Design; Applications; and Challenges"*, ICDCS Workshops 2004: 298-303.

[Romig1995] P. R. Romig and A. Samal: *"DeViouS: A Distributed Environment for Computer Vision"*, Journal of Software-Practice and Experience, vol 25, #1 Jan 1995.

[Schlögl2001] T. Schlögl, B. Wachmann, W. Kropatsch, H. Bischof: *"Evaluation of People Counting Systems"*, Proc. Workshop of the AAPR/ÖAGM; pp. 49-53; Berchtesgaden, Germany.

[Selker2000] T. Selker, W. Burleson: *"Context-Aware Design and Interaction in Computer Systems"*, IBM Systems Journal 39, Nos. 3&4. (6-2000).

[Siagian2004] C. Siagian and L. Itti: *"Biologically-Inspired Face Detection: Non-Brute-Force-Search Approach"*, In: First IEEE-CVPR International Workshop on Face Processing in Video, pp. 62-69, Jun 2004.

[Smith1995] S.M.Smith and J.M.Brady: *"SUSAN - A New Approach to Low Level Image Processing"*, Robotics Research Group, Department of Engineering Science, Oxford University, Technical Report TR95SMS1c, 1995.

[Soto2002] A. Soto: *"A Probabilistic Approach for the Adaptive Integration of Multiple Visual Cues Using an Agent Framework"*, doctoral dissertation, tech. report CMU-RI-TR-02-30, Robotics Institute, Carnegie Mellon University, October, 2002.

[Spengler2000] M. Spengler: *"Self-Organized Sensor Integration Applied to Multi-Modal Head Tracking"*, MSc Thesis, ETH Zurich, 2000.

[Stillman1998] S. Stillman, R. Tanawongsuwan and I. Essa: *"Tracking Multiple People with Multiple Cameras"*, Submitted (July 1998) to PUI 1998 Workshop, in conjunction with UIST 1998, San Francisco, CA. Nov. 1998.

[Svoboda2005] T. Svoboda, D. Martinec, and T. Pajdla: *"A convenient multi-camera self-calibration for virtual environments"*, PRESENCE: Teleoperators and Virtual Environments, 14(4), August 2005. To appear. MIT Press

[Terrillon2002] J.C. Terrillon, A. Pilpre, Y. Niwa and K. Yamamoto: *"Analysis of Human Skin Color Images for a Large Set of Color Space and for Different Camera Systems"*, Nara-ken New Public Hall, Nara, Japan, 2002.

[Vincent2002] E. Vincent and R. Laganiere: *"An Empirical Study of Some Feature Matching Strategies"*, in Proc. 15th International Conference on Vision Interface, pp. 139-145, Calgary, Canada, May 2002.

[Viola2001] P. Viola and M. Jones: *"Rapid object detection using a boosted cascade of simple features"*, Proc. IEEE Conf. on Computer Vision and Pattern Recognition, 2001.

[Walter2000] J. Walter, J. L. Welch, N. M. Amato: *"Distributed reconfiguration of metamorphic robot chains"*, Proc. of the 19th Annual ACM Symposium on Principles of Distributed Computing (PODC'00), 2000, p. 171-180.

[Want1992] R. Want, A. Hopper, V. Falcao, and J. Gibbons: *"The active badge location system"*, ACM Transactions on Information Systems, 10(1):91–102, January 1992.

[Weiser1991] M. Weiser: *"The Computer for the Twenty-First Century"*, Scientific American, pp. 94-10, September 1991.

[Wellner1993] P. Wellner: *"Interacting with paper on the DigitalDesk"*, In Com. of the ACM. 36(7), pp. 86-96, 1993.

[Whitaker1995] R. Whitaker: *"(Re-)Design of self-organizing enterprises"*, Workshop on "Groupware for Self-Organizing Units - The Autopoietic Turn in Organization Science and its Relevance for Groupware", ECSCW'95 (Fourth European Conference on Computer-Supported Cooperative Work), Stockholm (1995).

[Wikipedia] www.wikipedia.org

[Wolf2002] J. Wolf, W. Burgard, H. Burkhardt: *"Using an Image Retrieval System for Vision-based Mobile Robot Localization"*, in Proc. of the International Conference on Image and Video Retrieval (CIVR), 2002.

[Yanai1998] K.Yanai and K.Deguchi: *"An Architecture of Object Recognition System for Various Images Based on Multi-Agent"*, Proc. the 14th International Conference on Pattern Recognition, Australia, August 17-20, 278/281, 1998.

[Yang2002] M-H Yang , D.J. Kriegman , N. Ahuja: *"Detecting Faces in Images: A Survey"*, IEEE Transactions on Pattern Analysis and Machine Intelligence, v.24 n.1, p.34-58, January 2002.

[Zhang2000] Z. Zhang: *"A flexible new technique for camera calibration"*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 22(11):1330-1334, 2000.

[Zheng1999] Z. Zheng, H. Wang and E. Teoh: *"Analysis of Gray Level Corner Detection"*, Pattern Recognition Letters, 1999, Vol. 20, pp. 149-162.