WIRELESS SENSOR NETWORKS FOR SPECTRAL CAMERA CALIBRATION

BY

SUNAYANA SAHA

M.Sc. (Tech) Birla Institute of Technology and Science, 2001

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2003

Urbana, Illinois

Abstract

This work presents a unique spectral camera calibration technique using wireless "smart" micro electro-mechanical (MEMS) sensors. The word "smart" denotes capabilities of MEMS other than sensing, for instance, computing and communication. We foresee the use of widely distributed and deeply embedded "smart" MEMS sensors as potential calibration gauges for spectral cameras, such as, thermal infrared (IR) or visible spectrum cameras. Thus, the primary motivation of our presented work is to investigate the problem of spectral camera calibration in an indoor environment using the "smart" MEMS sensors. The main application of our work is in hazard-aware environments, or in any smart and intelligent spaces. There are also many other applications of advanced computer vision and robotics that try to automate manufacturing processes and can benefit from the use of "smart" MEMS and the results of our work.

We have investigated an optimal design of a spectral calibration system with thermal IR and hyperspectral cameras and the "smart" MEMS sensors known as the MICA sensors. The MICA sensor hardware is manufactured by Crossbow Inc. and programmed using TinyOS, an open source operating system, developed by the University of California at Berkeley. Our experimental results demonstrate a calibration of the thermal IR camera manufactured by Indigo System Corporation with MICA temperature sensors and our preliminary calibration of MICA luminance sensors using a hyperspectral camera manufactured by Opto-Knowledge Systems Inc.

In this thesis, we provide an overview of the features and challenges of the MEMS sensors in general, as recognized by current research efforts. We then propose a robust calibration system design by maintaining two main objectives – minimizing the wireless loss of data transmitted by the sensors and maximizing the information content of the collected data. The first objective is fulfilled by experimentally determining an optimal sensor network that gives the least data losses. Maximization of information content is achieved by synchronizing, registering and calibrating acquired 2D spectral camera images with precise point measurements obtained wirelessly from the MICA sensors. We believe that the reported outcomes of our experimental results will help other researchers designing optimal spectral camera calibration systems in future.

To

My dear husband, Abhishek,

&

family

Acknowledgements

I would like to sincerely thank my advisor, Dr. Peter Bajcsy, for his guidance and support. I am grateful to him for always being enthusiastic about my progress, encouraging me to try out new ideas and taking time out of his busy schedule when I needed his advice. I extend my gratitude to Rob Kooper, a research programmer at the National Center for Supercomputing Applications (NCSA), for offering suggestions during this work. Sang-Chul Lee, a Ph.D. student and an officemate at NCSA, helped me understand the software he developed that I used to assist me in my work. Finally my family and friends have been a constant source of support and an invaluable asset to me.

Table Of Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

Sensor networks are an emerging area of wireless computing, made possible by recent advances in micro electro-mechanical systems (MEMS) and low power technology. Their capability to sense the environment, do intelligent computation and wirelessly communicate information, together with their small size and low cost make them an attractive choice for a large number of useful application domains. Motion detection in military fields, sound recognition for habitat monitoring and factory automation through inventory tracking are some of the future applications of MEMS sensors. This work focuses on a novel application of MEMS sensors - spectral camera calibration. A spectral camera is understood as any camera operating at one spectral wavelength or multiple wavelengths for example, a thermal infrared camera or a colored visible spectrum camera.

Spectral cameras are widely used in smart or intelligent spaces and our focus is on building hazard aware spaces. Analyzing and understanding the content of images acquired by these cameras helps in making intelligent inferences about the environment. One of the problems faced by hazard-aware spaces is the lack of tools to automate image content analysis. Spectral camera calibration is one of the many steps that lead to better image content understanding.

In general, calibration is a "refined form of measurement conducted to assign numbers that represent relationships among particular properties of a measurement system" [1]. Our focus is on calibration of spectral values that lead to better scene understanding from acquired spectral data. Our approach for indoor camera calibration uses MEMS sensors as spectral calibration gauges. The fact that these 'smart' sensors can be distributed in large numbers, efficiently sample the environment, and form a wireless channel for information exchange, allows us to gather billions of sensory point measurements without much human intervention. Thus MEMS sensors can become a promising component for a spectral camera calibration system. The MEMS sensors used for this work are the MICA sensors. The MICA hardware is manufactured by Crossbow Inc. [2] and programmed using an open source operating system called TinyOS [3] that was developed by the University of California at Berkeley.

The primary motivation of our presented work is to investigate the problem of designing a spectral camera calibration system in an indoor environment using the "smart" MEMS. One of the spectral cameras we calibrate is a thermal IR camera. The thermal IR camera is an Omega model. It is a long-wavelength (7.5-13.5 microns) uncooled microbolometer camera designed for infrared applications and manufactured by Indigo Systems Corporation [4]. We also investigate the problem of calibrating the luminance sensors on the MICA with a hyperspectral camera manufactured by Opto-Knowledge Systems Inc. [5] and based on liquid crystal tunable filters [6]. The hyperspectral camera generates 2D images with a large number of spectral bands corresponding to a set of selected

wavelength ranges. The spectral calibration problem is formulated and solved as the optimization problem of minimizing wireless information loss and maximizing information content.

1.1 Our Contribution

We devise a spectral camera calibration technique for better image content analysis. In this work, we calibrate a thermal IR camera using MEMS sensors such that we can assign accurate temperature values (in °C or Kelvin and Fahrenheit) to each and every pixel in the thermal image. Though we have worked specifically with the thermal IR camera, our technique is general enough to be used for other kinds of spectral cameras. The novelty of our work lies in using 'smart' MEMS sensors as spectral gauges for camera calibration. Our method needs continuous data collection from the MEMS sensors and camera only for a short period of time, after which the data analysis is done on a dedicated PC. We also give experimental results of a new method for calibrating the luminance sensors on the MICA motes using a hyperspectral camera. Wireless 'smart' MEMS sensors and spectral cameras are envisaged to be crucial components for building future hazard aware spaces. We believe that the use of our calibration technique will also find applications in many other application domains.

Moreover, we have built this system by carefully reviewing all its components. For example, the first step towards the goal of camera calibration is to collect sensor data wirelessly. We perform thorough experimental studies to find the most efficient way to deploy a wireless sensor network. Experiments are conducted to determine wireless data

loss with respect to sensor data collection schemes, sensor node layout schemes, number of nodes, the degree of interference from other wireless devices and the distances from the central data collection unit. Data is finally collected through the most optimal sensor network design. Statistical techniques like temporal and spatial averaging are used to reduce camera noise in the acquired images and to compensate for data losses during sensor data acquisition and wireless communication. Also, we synchronize the camera and the sensors in order to achieve more accurate results. It is our belief that the lesson learnt in this work will guide future sensor network application designs.

1.2 Thesis Outline

The thesis is outlined as follows. Chapter 2 formulates the problem of camera calibration. It explains our approach in building the camera calibration system together with the goals we have tried to achieve. We present the related work in Chapter 3. Chapter 4 describes in detail the system components – the spectral camera and the sensors. Chapter 5 presents the calibration system in its entirety, with the experimental results of the calibration in Chapter 6. Finally Chapter 7 summarizes our work and discusses possible directions for future research.

CHAPTER 2

Problem Statement and System Overview

This chapter formally defines the spectral camera calibration problem. We give an overview of our approach to solving the calibration problem.

2.1 Spectral Camera Calibration Problem Definition

The problem of spectral camera calibration using MICA sensors can be formulated as follows:

Given a set of temporally changing, spatially local (point), spectral property measurements (for example, temperature and luminance) obtained by MICA sensors and a set of temporally changing, two-dimensional (2D raster), spectral images viewing a subset of MICA sensors, find the mapping between 2D image raw pixel values and their spectral values in engineering units (°C or Kelvin for temperature, lux or footcandles for luminance). Furthermore, the MICA sensors should transmit data wirelessly to the data-collecting computer while the camera would be wired to the same computer.

2.2 Overview of Our Approach

In this section we describe our approach to spectral camera calibration. Following that, we define the objectives for a spectral camera calibration system.

2.2.1 Calibration Approach

In general, a solution to the stated problem first requires a calibration of MICA sensor measurements with a spectral gauge. Figure 2.1 shows the calibration dependency graph between the three sensing devices - a spectral gauge, spectral camera and spectral

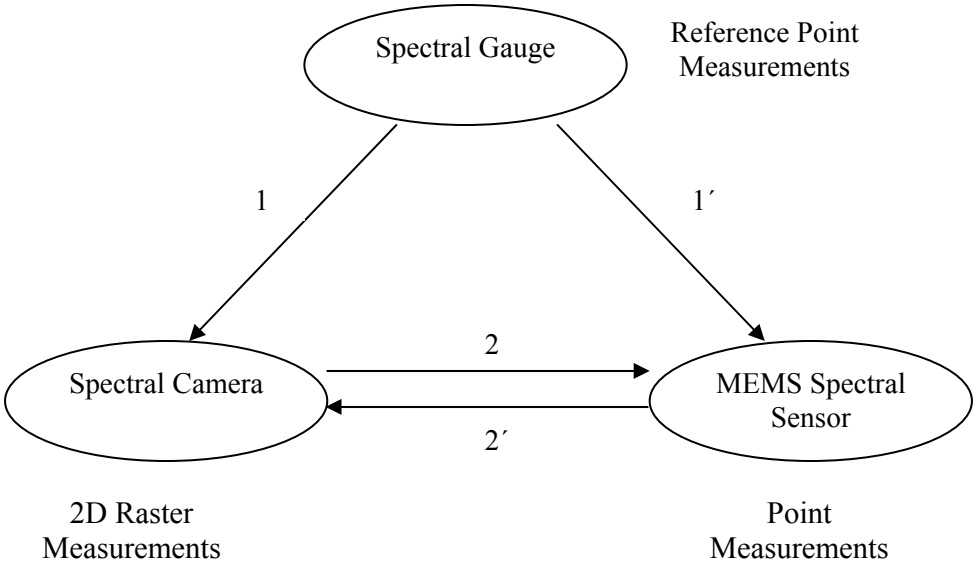


Figure 2.1 Calibration dependency graph.

sensors. The type of measurement each device yields is also specified next to the device. The spectral gauge (e.g. thermometer for temperature) provides measurements directly in engineering units (Kelvin or °C). According to Figure 2.1, there are two ways to achieve calibration. Using the spectral gauge, we first calibrate either the sensor (following arrow 1´) or the camera (following arrow 1). Once one of them is calibrated, we can calibrate the other one. In our work, we have addressed the problem of thermal camera calibration using the temperature sensors on the MICA motes. Thus, we calibrate the temperature sensors first and then calibrate the thermal camera using the sensors (1´ followed by 2´ in the graph in Figure 2.1). The latter part of our work consists of devising a calibration

procedure where we use a first calibrate a hyperspectral camera and then use it to calibrate the luminance sensors on the MICA sensors (1 followed by 2 in the graph in Figure 2.1). Due to similarity of most of the calibration process for the thermal camera and the luminance sensors, we only discuss the thermal camera calibration in detail. The calibration of the luminance sensors is discussed in Chapter 6 on experimental results.

A more elaborate procedure for camera calibration is shown in Figure 2.2 (a) and (b). The first step of calibration is depicted in Figure 2.2 (a). The MEMS sensors raw spectral measurements must be transformed to engineering units. The sensor readings from the sensors are collected wirelessly and stored on the PC. Also, each sensor is manually probed with the spectral gauge and the measurements of the spectral gauge are manually fed to the PC. The raw readings from the sensors and the measurements from the spectral gauge are then correlated to calibrate the sensors. This calibration step must be done manually and can be eliminated by using factory-calibrated sensors. In our spectral calibration approach, we consider the general case of un-calibrated point sensors, which is the case of MICA sensors. Once the sensors are calibrated, they can be used to calibrate the camera automatically as depicted in Figure 2.2 (b).

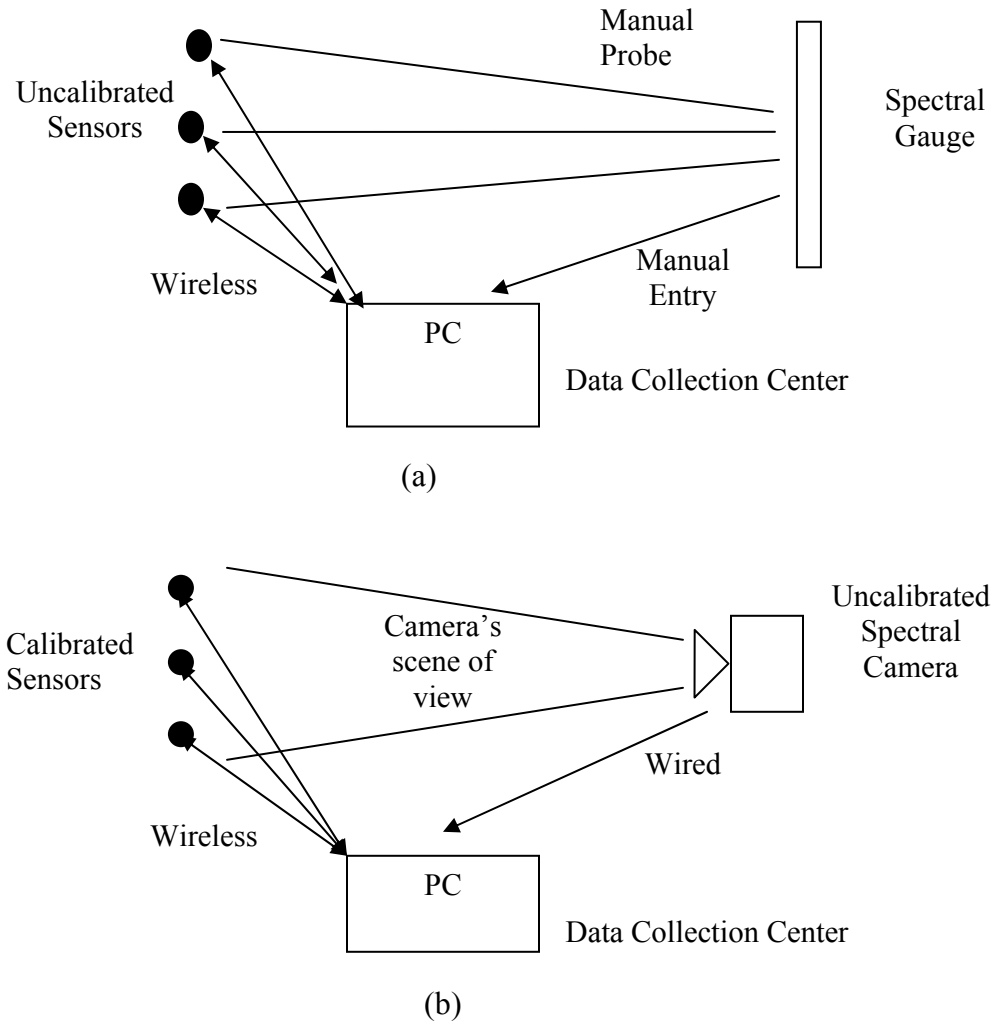


Figure 2.2 (a) Calibrating the MEMS sensors using the spectral gauge. (b) Calibrating the camera using the calibrated MEMS sensors

2.2.2 System Design Approach

Apart from solving the spectral camera calibration problem, our system design aims to achieve the following two objectives:

a) Minimize the information loss due to collisions during wireless data transmission between MICA sensors and the calibration station. Sensor network layout,

communication protocols and operational circumstances must be carefully considered to fulfill this objective.

b) Maximize the information content of the acquired raw data from the spectral camera and MICA sensors. Raw images and MICA sensor measurements have little value unless we can associate temporal or spatial attributes, such as, the time stamp or location to the measured spectral values.

Thus, in addition to finding the mapping between 2D raster and point measurements in engineering units, the spectral camera calibration system should minimize wireless information loss and maximize information content.

In this section we have given a broad overview of our spectral camera calibration system using the MEMS sensors. Since it is necessary to have a thorough understanding of the building blocks (sensors and cameras) of our system, the details of our system are presented in Chapter 5, after we have discussed our camera and sensors in Chapter 4.

CHAPTER 3

Related Work

This chapter discusses the previous work that is related to our camera calibration system. Since our system deals not only with camera calibration, but also with issues like information loss and synchronization, specific to sensor networks, we try to cover the related work both in the field of camera calibration as well as sensor networks.

3.1 Related Work on Camera Calibration

The problem of spectral camera calibration occurs in all application domains where automation using machine vision is highly desirable. For example, one can find the need for spectral camera calibration in the areas of remote sensing (radiometric and photogrammetric calibration of aerial [7] and satellite imagery), robotics (vegetation detection using near calibrated infrared and red wavelength imagery), astronomy (brightness estimation of stars using spectral imaging of the sky), military (battlefield analyses), smart spaces and any proactive computing applications. The main application driver for our work is the support of hazard aware spaces.

Camera calibration is performed for three main reasons. They are:

a) To rectify the distortion produced by the camera's optical components. Here, the calibration estimates the camera's intrinsic parameters like the focal length and

principal axis of transformation and the transformation between the image plane and the pixel coordinates.

b) To find the mapping between 3D and 2D projection. Extrinsic parameters are determined to get the location and orientation of the reference frame with respect to a known world reference frame.

c) To establish the mapping between the camera's raw pixel values and engineering units. The engineering units may depict a physical attribute like the temperature or luminance of the camera's field of view.

From the above list, (a) and (b) motivate most of the camera calibration work done in the field of 3-D computer vision. Knowledge of the camera's intrinsic and extrinsic parameters [8] is essential to construct 3-D structure or position of objects in real space, from 2-D images and also to remove distortions during image acquisition [9], [10], [11].

On the other hand, our work focuses on (c). By calibrating a spectral camera, we add sensing capability to the camera. For instance, associating temperature values to raw pixel values of a thermal IR camera image helps gain knowledge about the temperature of objects in the scene. This enhances the image's content understanding, for example, needed for discriminating hazardous and non-hazardous conditions.

Apart from the motivation (as explained above), our system differs from previous work in its approach. Our system deals with indoor spectral camera calibration. The previous work on indoor spectral camera calibration problem can be classified into approaches

based on (1) emitted (radiometric) or (2) reflected (photogrammetry based) energy/light using (a) black body or (b) reflectance gauges manually positioned in front of cameras and illuminated with specialized energy/light sources [12], [13]. We use the MEMS sensors (MICA motes) as our spectral gauges. The temperature sensors on the MICA motes provide us with point measurements that are then used to calibrate the thermal IR camera. In the case of outdoor spectral camera calibration, the approaches include additional considerations about atmospheric distortion, and passive and active type of sensing [14, Chapter 10], [15], [16]. We do not take into account these issues.

A very important application that can make use of our calibration system is a 'Severe Acute Respiratory Syndrome (SARS)' detection system. Being a contagious disease, airport authorities are trying to restrict movement of people who may have contracted SARS. One of the symptoms of SARS is high fever and many systems deployed in airports and other public areas try to track people with high temperature. Some recent commercial systems [17], [18] have proprietary software built to trace SARS. These use thermal cameras similar to ours to detect abnormal scene temperatures. However, they do not use the MICA sensors in their system.

To the best of our knowledge the use of 'smart' wireless MEMS sensors to develop an indoor spectral camera calibration system is novel. Our system is easy to deploy and does not require any additional components other than the MEMS sensors and the spectral camera, which we feel, will already be part of a hazard aware environment. By using the wireless capability of the sensors, we can use our system in remote areas, where

a far away central station can gather data from the deployed sensors through a multi hop sensor network. We believe that our work provides a unique foundation for a new spectral calibration approach.

3.2 Related Work on Sensor Networks

Sensor networks, like any other upcoming field, have many interesting challenges and have been the prime focus of many researchers. A lot of work has been done on transmission protocols for medium access control (MAC) [19], [20], [21] that take into consideration the limitations specific to sensor networks. By justifying that traditional MAC protocols (for example CSMA [22], [23]) are energy-inefficient for power-constrained sensor networks, these protocols suggest new ideas like sharing periodic sleep schedules among neighboring nodes [20], avoiding idle listening and overhearing problems [19]; and minimizing control messages [21]. However, we assume the presence of an underlying of MAC protocol to take care of lower level transmission details. We try to evaluate different design alternatives available at application level that affect the energy efficiency of the sensor network. We experimentally determine the energy efficiency of a network setup in terms of the percentage of data lost in it. Sensor network literature also proposes mechanisms for in-network data processing like 'data aggregation' [24] and 'directed diffusion' [25]. In a multi hop network these methods result in substantial energy savings by reducing the total number of transmissions and subsequently collisions. Our camera calibration system was built in an indoor laboratory where a single hop sensor network sufficed. In a single hop network, data aggregation is not required. Each node transmits data independently to the central processing and

logging unit, eliminating the need for complicated aggregation logic on sensor nodes. The aim of our study for designing an optimal sensor network is to measure the amount of data lost in a simple, single hop network in an indoor environment. In most other research efforts the sensors sense and transmit data either when queried or when an interesting event occurs. Thus, the network may be inactive (no transmissions) most of the times. In contrast, we focus on a network of sensors that continuously sense the environment and transmit data to the base station. Hence, our sensor network inherently generates much heavier traffic.

We have not come across a study similar to ours that would address application design issues under heavy traffic. Though we discuss time synchronization, we have not elaborated some on the topic in depth because many researchers have already devoted much effort on this area [26], [27].

CHAPTER 4

System Components

To fully understand the challenges of the proposed approach for our camera calibration system using the MEMS sensors, one must know the details of the two main components. This chapter delves into the details of the spectral camera and the MICA sensors we have used for our work.

4.1 Spectral Camera

The spectral camera we calibrated in this work is a thermal IR camera. It is a long-wavelength (7.5-13.5 microns) uncooled microbolometer Omega camera designed for infrared applications and manufactured by Indigo Systems Corporation, Goleta, CA [4]. It is controlled through the PC via RS232 serial port and the analog NTSC video output is digitized using a Hauppauge WinTV board [28]. The camera's operating temperature is from 0°C to 40°C. In its default setting the thermal camera gives grayscale images, as shown in Figure 4.1, showing a hot object (a hot water glass) as white while a cold object (a glass with ice cubes) as black.

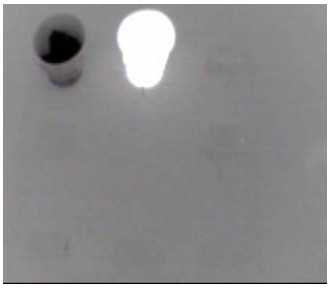


Figure 4.1 An image captured through the thermal IR camera

4.2 MEMS Sensors

In this section, we describe the MICA sensor hardware specifications, the software suite used to program the sensors and the unique communication paradigm designed especially for these networked sensors. Constraints related to MEMS specification are elaborated. We also enlist some prototype sensor network applications, followed by the common challenges and the proposed solutions for sensor network applications. This section is not specific to our camera calibration system. We conducted a literature review in the area of sensor network and this section gives a good overview of work actively pursued in the sensor network research community. The term sensor *'mote'* and sensor *'node'* are used interchangeably in this document. 'Mote' is a commonly coined term in MEMS sensor literature.

4.2.1 MEMS Hardware

The hardware for our particular MEMS sensor, the MICA sensor [29], was manufactured by Crossbow Technology Inc. [2]. A MICA sensor consists of a MICA mote processor/radio (MPR) board and a sensor board (like the MTS101CA [30]). Figure 4.2 shows an MPR board. Figure 4.3 is a block diagram of the board shown in Figure 4.2.

Figure 4.2 A MICA MPR board.

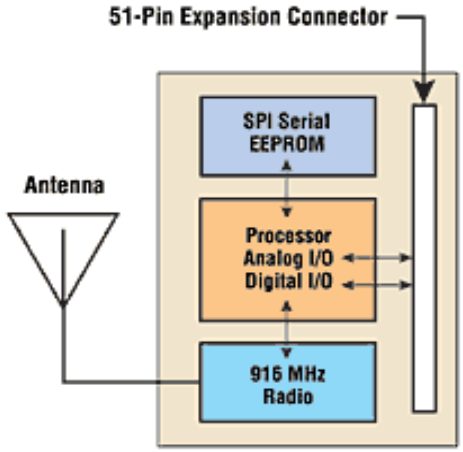


Figure 4.3 Block diagram of a MICA MPR board.

The MICA MPR board specifications are:

- A 51-pin expansion connector to attach a sensor board.
- 4MHz Atmega 128L processor.
- 128K bytes Flash, 4K bytes SRAM and 4K bytes of EEPROM.
- 916MHz radio transceiver with a maximum data rate of 40Kbits/sec.
- Attached AA (2) battery pack for power supply.

- An analog-to-digital converter (ADC) converts the analog signals from the different sensors on the sensor board to 10-bit digital format.

We used the MTS101CA [30] series sensor boards. They have a precision thermistor and a luminance sensor. The sensor board is attached to the MICA programming board through the 51-pin expansion connector. To develop an application, the EEPROM can be programmed using TinyOS [3], an open source operating system developed at the University of California, Berkeley. TinyOS is discussed in detail in the following section. The sensor board together with the programmable MICA board forms a device capable of sensing, computation and communication.

A MICA mote may cost (depending on the set of sensors attached to it), around 100 – 300 dollars and is about 4 inch by 2 inch in size (though ongoing research is aiming at smaller and cheaper devices). The computational, communication and sensing features together with the low cost and size make these MEMS sensors a very attractive alternative to their traditional counterparts. However, all the above hardware capabilities have severe limitations. Sensors motes have limited processing and storage capability. Moreover, since AA batteries supply power to sensors, energy becomes an expensive commodity.

Keeping energy conservation in mind, the processor has three sleep modes: 'idle' which just shuts the processor off; 'power down', which shuts everything off except the watch-dog; and 'power save', which is similar to power-down, but leaves an asynchronous timer

running [31]. Power consumption equates to battery life. Long battery life is desired, and in some applications, one to five years is required. The processors, radio, and a typical sensor load consume a power of about 30 μW when in sleep mode and 100 mW in non-sleep mode. Going by these numbers, most sensor networks being developed today adopt the philosophy of getting work done as quickly as possible and then going into sleep mode. Apart from putting the sensors in sleep mode, there are other ways to minimize loss of energy, one of them being, reducing collisions during wireless transmissions. We optimize energy usage in our application by designing a network that minimizes losses through collisions (explained in later sections).

The above hardware related constraints heavily influence any sensor application design and will be discussed in detail in later sections.

4.2.2 MEMS Software

In order to build sensor applications the MICA motes must be programmed with TinyOS [3] code. TinyOS is an open source software platform developed by researchers at UC Berkeley and actively supported by a large community of users. It is a small operating system that allows networking, power management and sensor measurement details to be abstracted from core application development [31]. TinyOS is optimized for efficient computational, energy and storage usage. The key to TinyOS's functionality is the NesC (network-embedded-systems-C) compiler, which is used to compile TinyOS programs. NesC has a C like structure and provides several advantages such as interfaces, wire error detection, automatic document generation, and facilitation of significant code

optimizations. One of its major goals is to make TinyOS code cleaner, easier to understand, and easier to write.

TinyOS consists of a tiny scheduler and a graph of components [31]. A component has a set of *'command'* and *'event'* handlers. An event is like an interrupt sent from one component to another informing that something of interest to the receiver has occurred. For example, once an application's component has set the Timer component for a particular interval, the Timer component will inform the application through the 'Timer.fired()' *'event'*, when the specified interval is over. Conversely, a component can invoke another component's functionality using the callee component's *'command'* interface. An application would be using the command 'setTimer()' of the Timer component to use the timer functionality. Figure 4.4 explains the interactions between any two TinyOS components.

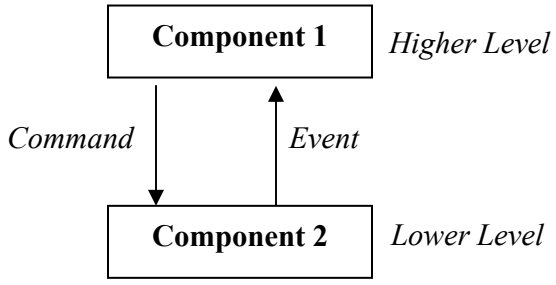


Figure 4.4 Interaction between TinyOS components.

In general, components interact by receiving commands from 'higher level' components and handling events from 'lower level' components.

This clean abstraction allows rapid code development. Application development then involves plugging your components with the already implemented components - specifying how they will be linked to each other, that is, which component will invoke what command and which component will signal another about what events. This linkage is specified in a configuration file, which is one of the two files for any TinyOS component. The other file is the module file that contains the actual code for an application.

TinyOS events are non-preemptive and are always run to completion. Thus, they must only do a small amount of work before completing otherwise they will stall the rest of the processing. In order to perform longer processing operations, TinyOS introduces a scheduling hierarchy consisting of *'tasks'* and *'events'*. Tasks are used to perform longer processing operations, such as background data processing and can be preempted by events. However, one task cannot preempt another task. Thus, a task should not spin or block for long periods of time. If one needs to run a series of long operations, one should have a separate task for each operation.

While writing an application code, care must be taken in terms of power, CPU and memory usage. One should avoid writing too complex programs. Assuming unlimited memory can lead to grievous consequences. The TinyOS architecture and scheduling policies must be kept in mind to write 'efficient' and 'correct' code. Careful planning and coding will lead to better application development and deployment.

4.2.3 Communication Paradigm

One of the most attractive features of the MICA sensors is their wireless capability allowing them to form a network of sensors. The storage, processing and energy limitations presented above demand an entirely different communication model as compared to legacy communication models like TCP/UDP. TCP [32] like protocols require huge buffers that are prohibitive for sensors with limited storage. Most traditional protocols are blocking in nature since they use polling to receive data. This is too wasteful for the MICA sensors since sensors are engaged in multiple interactions with the external environment – sensing and communicating, thereby needing support for concurrent operations and agility. As most sensor applications are event driven, researchers at UC Berkeley have designed an event based 'Active Message' communication model [33] for the MICA sensors. This model avoids the busy-waiting or polling concept to support the concurrency intensive nature of networked sensors while allowing for efficient modularity. Moreover, the event-based nature of this communication protocol fits well with the event-based TinyOS operating system that runs on these motes (explained in the previous section). Essentially, 'Active Message (AM)' paradigm combines communication with computation, while matching the communication primitives to hardware capabilities. It can be viewed as a distributed event model where networked nodes interact through messages. Each 'Active Message' contains the name of a user-level handler that is invoked on the target node upon arrival and a data payload that is passed to the handler as its arguments. The target node must register the type of message it is interested in receiving. Internally, on receiving a packet, the Active Message component on a node first checks the address of its node with the one

specified in the packet. If there is an address match and the handler in the packet has been registered by an application on the receiver, then the handler function is invoked with the data payload passed in as its arguments.

The 'Active Message' component uses the underlying packet-processing components to actually send or receive messages over the radio or serial port. Figure 4.5 depicts how the 'Active Message' component fits with the rest of the components for a generic MICA application built on TinyOS.

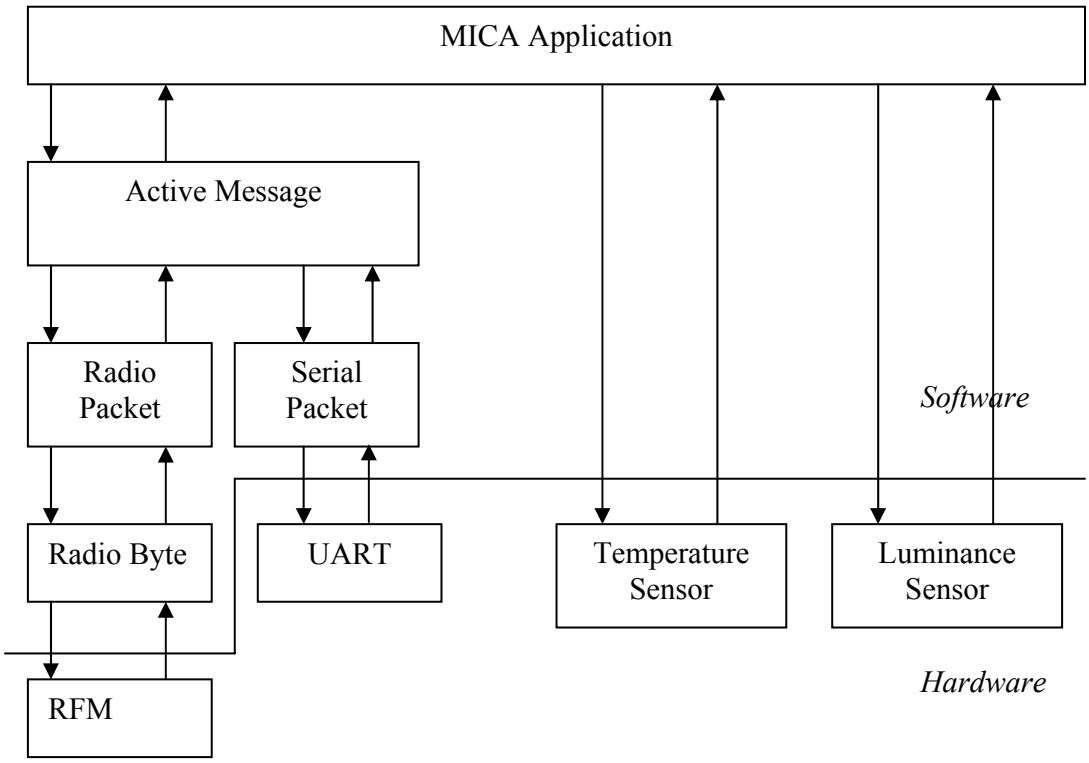


Figure 4.5 Component graph for a MICA application with the Active Message component.

The ‘Active Message’ paradigm overcomes the unnecessary waste of energy and loss of concurrency that polling based communication models have. Moreover it easily fits into the event model that TinyOS supports.

4.2.4 Prototype Applications

Sensor networks are revolutionizing the information-gathering process. Their low costs and small size make it possible to deploy them in large numbers, in inhospitable physical conditions such as remote geographical locations, toxic areas or less accessible places like machine interiors. The primary purpose of sensor networks is to sample the environment for sensory information and propagate this information wirelessly to bigger processing units or base station. There are many applications for wireless sensor networks [34], [35]. Some are new; others are traditional sensor applications that can be improved using wireless sensors. Some applications under current survey include:

a) Physical security for military operations – motion detection through accelerometer sensors for enemies’ army movement tracking in a battlefield.
b) Environmental data collection – for example, habitat monitoring [35] and bird/animal call recognition [34]. Sensors are programmed with waveforms of bird and animal calls of interest. Sensors sense the environment and compare a bird/animal call with the one of interest. If there is a match, this information is passed wirelessly to the central processing unit.
c) Seismic monitoring using magnetometer sensors.
d) Industrial automation – sensors could keep track of the inventory levels and send signals to the base station if the inventory level goes below a specific level [36].

e) Future consumer applications, including smart homes. Sensors in such applications could help in detecting anomalous situations, or could simply be used to link different household devices.

Though all these applications seem very appealing and useful, they are still in the early stages to development. Many challenges in wireless sensor networks have to be overcome in order to make the above applications successful. The next section points out the difficulties encountered by the sensor network research community.

4.2.5 Sensor Network Challenges

In this section we will enumerate the challenges faced by any application in wireless sensor networks and discuss the solutions proposed in the sensor networks literature to work around these problems. The solutions vary from low-level wireless networking decisions to high-level application design specifics. The proposed solutions form the foundation of any robust wireless sensor application.

4.2.5.1 Power Constraints

A good overview of the issues that must be kept in mind while designing sensor networks is provided in [37]. Power remains to be the most valuable resource for a functional mote. Without power, everything, from communication, sensing to computation will come to a standstill. Energy efficiency must be strived for at every level of the system design – from power-aware application level code, to network routing decisions and range of wireless transmissions. An application can achieve energy efficiency through careful

coding. Putting the CPU to sleep when there is no work (as discussed earlier) is one energy-preserving concept. However, most sensor applications place the nodes in unattended environment. Operating on a 3V power source, typically two AA batteries, sooner or later the batteries will run out of power, making its mote ineffective. Since, loosing a mote would mean loss of information from that part of the sensory field, it is important to introduce redundancy in the network. Even if one mote dies out, others should be able to keep the network up and running. Redundancy is also crucial due to a mote's susceptibility to failure. It is hoped that one day, these motes would become so inexpensive, that discarding them, once they become inoperable due to power dissipation or failure, will be a cheaper alternative than repairing them. Along with redundancy, one needs a self-configuring network infrastructure. The network routing protocols should be such that the failure of one intermediate forwarding node should not deter other motes from communicating with the central station. Thus, most communication protocols [25] developed for sensor networks aim for zero-configuration and zero-maintenance of sensor nodes.

Power is consumed by the radio for receiving and transmitting messages. Transmission failures due to collisions in the network result in power wastage because the motes have to retransmit the same message until it is successfully delivered. Therefore, what are required are routing strategies and a network stack that save power by minimizing collisions. There are numerous protocols to minimize energy wastage in message transmission. PAMAS [19] and S-MAC [20] are some commonly cited protocols that are variants of the MACAW protocol (used in traditional wireless networking domain).

Traditional networking protocols require handshaking mechanisms between the sender and the receiver before the actual data transmission. They also require motes to listen to the channel and only use it if the channel is free. Both these techniques are a drain on precious energy. Sensor network protocols suggest careful and limited use of handshaking, as transmissions are very energy intensive. They also suggest minimizing idle listening. A mote should put itself to sleep if it finds that the channel is busy, since the mote cannot transmit anyway. Smaller packet sizes are also advocated for sensor network applications. Larger packets have a greater probability of collisions and corruption leading to energy wastage. Another idea is to deploy multi hop instead of single hop networks. Single hop networks, where messages go from the source and the sink directly, have the difficulty that devices far from the information sink require more energy to report data [37]. Multi hop networks reduce the overall energy required to transmit a message. However, sometimes multi hop networks are avoided since they are too complicated for the task in hand and the throughput also goes down as the number of nodes increases [38].

Another concept that can be exploited in the sensor network is *'data-aggregation'* [24], [25], [39]. Most sensor networks are application specific and intermediate nodes are aware of the semantics of the data content they forward. Thus, instead of each mote sending its own packet, intermediate motes can aggregate (e.g. average all packet values or do duplicate detection and suppression) packets from their neighbors thus reducing the number of transmissions flowing towards the final destination. Reduced transmissions mean decreased possibility of collisions, which in turn gives energy savings.

In short, we need robust and energy aware routing protocols and application designs. Applications should be able to tolerate some amount of failure. They can extend the lifetime of motes using techniques like data aggregation, at the cost of some delay introduced by the processing at the intermediate nodes.

4.2.5.2 Memory and CPU Constraints

Memory and CPU constraints may limit the capabilities of an application. Instead of allowing large amounts of data to be stored in sensor motes, applications should let motes forward small amounts of data towards the base station or the central processing unit. Smaller software footprint (fewer lines of application code) reduces the demands on scarce memory. Some research papers suggest that we should avoid using the motes for complex computations, since an idle CPU can be put to sleep. The motes should primarily collect information from the environment and do some minimal amount of calculations, like filtering of uninteresting or duplicate data. A more powerful central processing station must handle computationally intensive tasks.

In this chapter we have given the specifications of the main components of the spectral camera system. We have described in detail the hardware, software and communication model for the MICA motes together with the common challenges faced by sensor network applications. It was important to build this background so as to understand better the reasons behind various design choices in the next chapter.

CHAPTER 5

Camera Calibration System

Now that we know the specifications of the camera and the MICA sensors, we are well equipped to get into the specifics of our camera calibration system. In Chapter 2 we had defined the problem of camera calibration and had mentioned the other two main objectives of our system – maximizing information content and minimizing wireless data loss. This chapter describes our system setup and the methods we follow to achieve the system's desired end results. Though this chapter emphasizes on the calibration of the Omega thermal camera using the temperature sensor on the MICA MTS sensor board, other cameras can be similarly calibrated. For example, the hyperspectral camera can be calibrated using MICA luminance sensor sensing at a narrow band wavelength. We believe that our method provides a foundation for a new spectral calibration approach.

5.1 System Setup

We carried out our camera calibration experiments in an indoor laboratory. The calibration system components are set up as follows:

a) Sensors: We have used nine MICA sensors in our system. One of the sensors is attached to a PC through a serial port. This sensor acts as the base station, collecting data wirelessly from other sensors and forwarding it to the PC using the serial port. The other eight sensor motes continuously sense the ambient temperature and record a point temperature value after every 100 milliseconds.

After every 100 milliseconds, the sensor mote would get the raw temperature value through its analog-to-digital-converter (ADC) and store it in a local array. The local array size was fixed to hold only 10 readings. We chose to store a maximum of 10 readings on a mote because of our concern for limited memory on the motes. Also, owing to the fact that a larger packet has a greater chance of getting corrupted through collision, we preferred transmissions of smaller packet sizes. Since our camera calibration experiments were confined to a relatively small laboratory, we used a *'single hop'* sensor network to relay data from a sensor and the base station. This helped us to avoid the complications in a multi hop network and concentrate on the core issue of calibration. Depending on the scheme used for data collection (discussed later), a sensor would send a packet containing 10 readings to the base station sensor mote. The base station mote will forward the data to the PC that will save this data to file for interpretation and analysis later.

b) Camera: The Omega thermal camera is attached to the PC through the Hauppauge WinTV board [28] that digitizes the images. Image acquisition is controlled programmatically using Java Media Framework (JMF) [40] libraries. Once images capture starts, images are acquired and saved continuously until the data acquisition is stopped.

c) Auxiliary spectral gauge: as explained in Chapter 2, we need a third spectral gauge to calibrate one of the above two components. To calibrate the temperature sensors for thermal camera calibration, we used a regular thermometer used by chemists. It measured temperature directly in engineering units of °C and

provided temperature readings in the range [-40°C, 150°C] with reading uncertainty equal to ±1°C.

5.2 Calibration Procedure

In order to do the spectral calibration we follow the steps that were outlined in Chapter 2. As depicted in Figure 2 (a), we first calibrate the MICA sensors with the spectral gauge. Then we calibrate the spectral camera using the calibrated MICA sensors (Figure 2 (b)).

The steps involve:

a) MICA sensors are programmed to sense and send temperature readings to the PC, over a certain time period.
b) During the same time period, temperature measurements are collected with a thermometer (a calibration gauge).
c) A calibration transformation is established for MICA temperature sensors as a combination of factory recommended formula (given later) and reference point measurements taken from the thermometer. This step completes the calibration of the MICA sensors.
d) The calibrated MICA sensors are programmed to sense point temperatures, and the thermal camera is focused so that its scene of view contains the sensors.
e) Both thermal IR camera and MICA sensors are initiated to acquire data synchronously (described later). The MICA sensors send their readings over to the PC. At the same time, the thermal camera starts streaming in images to the PC.

f) The MICA sensor measurements are received and transformed into °C using the calibration formula derived in step (c).

g) MICA temperature sensor locations are identified in the thermal IR image, and the statistical sample means and standard deviations of the calibrated MICA sensor measurements and the thermal IR image pixel values at the MICA sensor locations are related to form the final calibration transformation. In this step, if the entire scene viewed by a thermal IR camera is temperature homogeneous then MICA temperature sensor locations in the thermal IR image do not have to be identified and the statistical sample mean and standard deviation for the thermal IR image can be computed over the entire image.

h) The last step is the spectral image calibration where each pixel of the thermal image is mapped to a temperature value by using a linear model of the calibration transformation obtained in the previous step.

5.3 System Design Objectives

During the execution of the previously described spectral calibration procedure, we strive to meet two objectives - maximization of data information content and minimization of wireless information loss. Information content is maximized by integrating the temporal and spatial information into the calibration process. Minimization of wireless information loss is attained via optimal MICA sensor network design. These two aspects are addressed in this section.

5.3.1 Maximization of Information Content

Maximizing information content can lead to increased accuracy of the end result. We achieved maximization of information content by obtaining temporal and spatial information about each measurement.

5.3.1.1 Temporal Information

In order to be able to correlate two or more measurements precisely, it is necessary to store temporal information about each measurement. We stored temporal information by synchronizing the sensors with each other as well as with the camera and time stamping sensor readings and the camera images. We briefly discuss the implementation of these two issues now. More details on the implementation can be found in [41].

5.3.1.1.1 Synchronization of Sensors and Camera

Synchronization has several important implications for our system:

a) Starting the sensors and the camera at the same time allows us to do temporal correlation of measurements collected from different sources.

b) Instead of a sensor starting to sense the ambience the instant it is switched on, it is a better design if sensors wait for a signal to start. The base station could signal the sensors to start, when it is ready to receive data from the sensors. This will lead to huge energy savings, as it will eliminate the scenario where the sensors are sensing the environment and transmitting data when the base station is not even interested in the data. Similarly, triggering the camera to begin image acquisition only when the sensors start, leads to space savings on the PC.

We handled the issue of synchronization by instructing the sensors to wait for a *'RESET'* signal from the base station. Until they receive this signal, they do not start sensing the environment. The base station broadcasts the *'RESET'* signal. Once this signal is received, a sensor starts sensing the ambience. At the same time when the base station sends the *'RESET'* signal to the sensors, it starts another thread in the software which triggers the camera to start continuous image acquisition.

Theoretically, the precision to which the nodes are synchronized with each other depends on the possible sources of synchronization message latency – send time, access time, propagation time and receive time [26], [27]. If we only consider propagation time, then inter-node synchronization can be estimated as follows:

Suppose the propagation delay of RESET message from the base station to a node is *'p'* time units. Then, if all sensors are placed at an equal distance from the base station, they will get the *'RESET'* message *'p'* time units after it is broadcast from the base station. Thus they will all be perfectly synchronized. If the sensors are placed at varying distances from the base station, they would get the RESET message between p_{min} and p_{max} time units, where p_{min} is the propagation delay to the closest sensor and p_{max} is the propagation delay to the farthest sensor. In this case, the nodes would be synchronized within a bound of *'*$p_{max} - p_{min}$*'* time units.

Another hurdle for synchronization is clock-drift. The hardware clocks on different nodes may count time at slightly differing rates. With progressing time, a particular sensor may be far ahead or behind some other sensor. This factor, together with the slight propagation delay of the synchronization message (as described above) can lead to a significant offset between different sensor timers, thus making them out of sync. A simple solution to rectify this asynchrony is by specifying an upper bound *'E'* for this asynchrony, and re-synchronizing the sensors at time *'t',* after which the error goes beyond *'E'*. If we know the clock drift rate *'d'* (a clock manufacturer specified value), propagation delays p_{min} and p_{max} and the error bound '*E',* calculating *'t'* is straightforward.

Let $t_{diff} = p_{max} - p_{min}$. In the beginning, a node may be $(t_{diff}(1 \pm d))$ time-units out of sync with another node. After *t* time units, the maximum time units a node may be out of sync with another node is $(t_{diff}(1 \pm d)) \pm (2td)$. Bounding this by *E,* where $0 < E < 1$, we get $|(t_{diff}(1 \pm d)) \pm (2td)| < E$. Solving to get an upper bound, we get $t < (E-(t_{diff}(1+ d)))/2d$. This gives the maximum value for *t* after which the nodes become asynchronous and need to be re-synchronized.

5.3.1.1.2 Time Stamping Measurements

Time stamping helps in useful data analysis. Once the components are synchronized with one another, one can correlate a set of time stamped measurements.

Sensors do not have any notion of clock time, which is available on bigger computational devices – like the day and time information available on our PCs. However, sensors do have a timer, which can be programmed to start at some instance and then repeat after a fixed interval. In order to time stamp a sensor's readings, we start the timer on each sensor when we are synchronizing the sensors with the *'RESET'* signal. The timer is set to repeat after every 100 milliseconds. Subsequently, we maintain a counter (called *'count'*) that is incremented by one, after every timer interval. This counter will give us the time relative to the timer start time. A counter value of 10 would thus mean that 10 * 100 = 1000 milliseconds have elapsed since the start of the timer. Moreover, since all the sensors start their timers at the same time (when they are synchronized) a counter value of *'c'* on two or more sensors would refer to the same time instance. This counter value is copied into the packet sent to the base station. We realize that sending the counter value wirelessly to the base station consumes bandwidth, but it is important to do so, so that we can correlate a set of readings. It also insures that the base station only records readings in increasing order of time and drops readings that appear late. Also, it helps in evaluating when readings were lost (described later) and compare readings taken at the same time instance on different sensor nodes. We do try to save bandwidth by transmitting a single counter value for all the readings in a packet instead of sending a counter value for each reading in the packet. We are able perform the aforesaid bandwidth saving because of the specific manner in which we programmed our sensors. We implemented our time stamp logic as follows:

As per our setup, we maintain 10 *'consecutive'* readings in the local array on a sensor, before sending a packet with those 10 readings to the base station. As soon as a packet is transmitted, the local array is emptied out. While recording the first value into the array, the current *'count'* value is stored in a separate variable, say *'saveCount'*. While preparing a packet to be sent to the base station, *'saveCount'* is copied into a counter field in the packet. This field gives a timestamp for each value in the packet. To understand this better, let us suppose that a packet has not been sent yet. When the first reading of that packet is being stored in the sensor's local array, we would note the counter *'count'*s current value, say '*v*', into *'saveCount'*. If the timer interval is 100 milliseconds and if '*v*' is 100, it means that the first reading of the packet was recorded 100 * 100 = 10 seconds after the start of the sensor mote's timer. Since a packet contains contiguous readings, the next reading will automatically be 101 * 100 = 10.1 seconds after the start and so on.

The camera images are time stamped by using the CPU clock on the PC. Just before saving an acquired image to disk, the time that has elapsed since the camera was started (and synchronized with the sensors), is attached to the image filename. Later, while correlating the image and sensor readings, an image's timestamp is matched with the timestamp of a reading.

5.3.1.2 Spatial Information

The last step of camera calibration needs to identify the MICA temperature sensor locations in the thermal IR image. The thermal pixel values of these locations are statistically sampled to attain the final calibration. If the entire scene viewed by a thermal

IR camera is temperature homogeneous then MICA temperature sensor spatial locations in the thermal IR image do not have to be identified and the statistical sample mean and standard deviation for the thermal IR image can be computed over the entire image. This method is straightforward. However, in our experiments, we found that the scene temperature was not homogenous according to the thermal image. This necessitated adding spatial information to the calibration process.

The process of identifying MICA temperature sensor locations in a thermal IR image can be achieved in several ways depending on the available position information. Position information availability about MICA sensor and thermal IR camera positions and orientations can be known a priori, sensed and estimated, or unknown. We describe our approach for the last scenario (information is unknown) by performing a geometric calibration of thermal IR camera and MICA sensor locations. Figure 5.1 shows a thermal image taken during the calibration process.

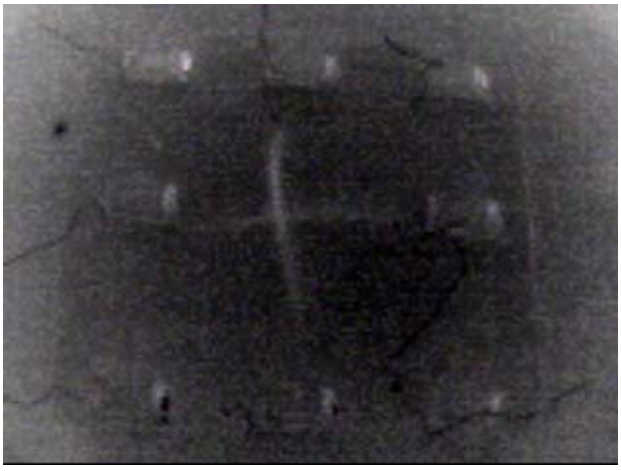


Figure 5.1 A thermal image captured during calibration.

As is obvious from Figure 5.1, the difficulty with the geometric sensor calibration is in the fact that it is very hard to see any salient features of temperature sensors in thermal IR images because their temperature is very close to the ambient temperature. In our approach, we chose to identify the MICA temperature sensors in a visible spectrum image since many salient MICA temperature sensor features are noticeable in the visible spectrum images. This approach is also used, for example, in remote sensing area [42]. Thus, the geometric calibration is accomplished by:

a) Acquiring visible spectrum images in sync with thermal IR images.
b) Identifying MICA temperature sensor locations in the visible spectrum images.
c) Registering visible and thermal IR spectral images to obtain accurate temperature sensor locations in a thermal IR camera coordinate system.
d) Using the spatial information to find pairs of point and raster measurements.

Figure 5.2 shows a visible spectral image that was registered with the image in Figure 5.1 to obtain spatial information about the temperature sensors. The tiny black spot in the

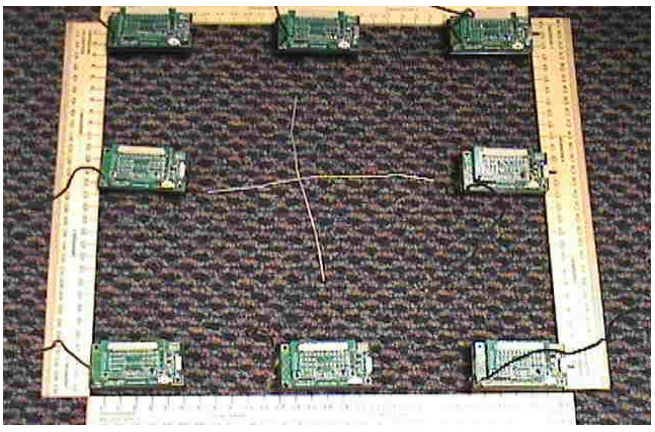


Figure 5.2 Experimental setup viewed through a visible spectral camera.

middle (slightly towards the right) of a sensor mote is the thermistor, whose location in the thermal image we need to determine.

The image analysis tools in the "Image To Knowledge (i2k)" software suite [43] were used to register the two images. In order to register two images, we have to select at least 3 points in the thermal IR and visible spectrum images and construct an affine transformation matrix. Since the experimental setup consisted of only sensors that appeared blurred in the thermal IR image, we also placed two heated metallic wires in field of view of the cameras in addition to the MICA sensors. These two wires appear as white lines in the thermal IR image in Figure 5.1 and can be clearly seen in the visible spectrum image in Figure 5.2. Consequently, while registering the two images, we match the end points of these wires in both the images.

5.3.2 Minimization of Wireless Information Loss

Wireless information loss occurs due to collisions during data transmission between the sensors and the base station. To minimize the wireless losses, we evaluate our sensor network by varying different network parameters. These parameters include data collection schemes, number of active motes, spatial arrangement of sensor motes, node distance from the base station and interference from other wireless devices. A network setting is considered better than another if it gives lower data losses. By combining the results of all the experiments, we design an optimal sensor network.

5.3.2.1 Determining the Amount of Data Lost

While comparing two or more sensor network setups, we take into account the data loss in each of them. We believe that apart from the data lost during initial startup and stabilization at each sensor node, all other data losses are attributed to collisions during transmission. A packet corrupted during transmissions is discarded at the base station.

All data losses are calculated on the PC that finally logs the data from all the sensors. A program on the PC starts tracking the losses as soon as the base station sends the *'RESET'* message to synchronize and start the sensor nodes. Data loss is measured in terms of the total number of missing readings, from all the sensor nodes. The counter value used to time stamp a set of readings in a packet (explained earlier in 5.3.1.1.2) is used to detect data losses. Detecting losses is based on the following observation:

If, on the PC, the previous packet from mote *'a'* had a counter value of *'*x*'*, then the next packet from *'a'* should have a counter value of *'x + 10'*, since the previous packet contained 10 readings. If however, the next packet counter *'y',* is greater than *'x + 10'*, then the readings for counter values between 'y' and 'x + 10' are missing i.e., 'y – (x + 10)' readings are considered lost.

In short, the readings lost are determined by observing the counter value in a packet with respect to the previous packet's counter value from the same mote. A gap in the counter value could be due to a packet lost in transmission, readings lost on a mote itself due to factors like stabilization during initial startup. From our present understanding of the

TinyOS radio and network stack, we believe that if a packet gets corrupted, then all readings are dropped. Thus, readings lost due to packet corruption through collisions will be a multiple of the number of readings in a packet (10 in our case). However, readings lost on a mote need not be a multiple of the number of readings stored in a packet.

At the end of the experiment, the data loss is calculated as a percentage of missing readings, which is the ratio of the total number of missing readings from all the motes, to the total number of (missing + correct) readings from all motes.

5.3.2.2 Network Parameters

Different parameters in a network setup can lead to varying amounts of data loss. The main parameters that we considered to find the most efficient network setup are:

a) Data collection schemes: For our application of camera calibration, we required a simple data collection approach that would ensure the least amount of data loss in transmissions to the base station. We thus evaluated two different mechanisms for collecting data from the sensors. They are:

- *Autosend* scheme: in this scheme, a sensor would send a packet to the base station as soon as it had 10 readings in its local array. Ideally, a sensor would transmit a packet for the temperature readings and a packet for the luminance readings after every one second (100 milliseconds/reading * 10 readings/packet = 1 second/packet).
- *Query* scheme: In this scheme, a sensor still continuously senses the environment, but it does not send data as soon as it has the fixed number of

readings in its local array. Instead, the base station queries each sensor mote in a round robin fashion. When a mote receives a query message from the base station, it checks to see if a packet with 10 readings can be sent. If yes, it is sent immediately. Otherwise, it simply sets a local flag indicating that the base station query is pending. When 10 readings have been collected, if the base station query is pending, a packet with the readings is sent. There is a catch in this scheme. If a mote does not have 10 readings when the base station queried it and it sends the packet later, when the 10 readings were ready, then it may collide with another mote's transmissions. However, we try to limit such a scenario by making the base station wait for a queried node's response for a finite amount of time, before it queries the next sensor. It is hoped that a mote would have 10 readings by then.

There are reasons why we choose the above two schemes. The *'autosend'* scheme is appealing because of its simplicity. Each node works as an independent unit and transmits a packet to the base station whenever a packet worth of data is ready. The base station too has no other responsibility than to collect and log data flowing towards it. However, since there is no control on any node's transmissions, there are bound to be collisions. That is the reason why we came up with a contrasting scheme – the *'query'* scheme, where we can control a node's transmission to a certain extent.

b) Number of sensor nodes – the number of nodes in the network is increased from 1 to 7 nodes. This would tell how data loss is affected by increased traffic in the network.

c) Spatial Arrangement of the sensors. There were three mote arrangements that we tried to compare and evaluate. They were:

- Nodes arranged in a *'straight-line'* and within 10 to 15 inches from the base station and about 3 inches from each other.
- Motes arranged in a *'circular'* fashion around the base station with a radial distance of 10 inches between the base station and a mote. This is an arrangement where all nodes are equally close to the base station.
- Nodes scattered in a *'random'* fashion in the laboratory – at a distance anywhere from 10 inches to 150 inches from the base station.

c) Distance from the base station – a mote's distance was increased gradually up to the maximum range of transmission (about 70 feet).

d) Interference from other devices – the sensor node's transmission was tested in the presence of other wireless devices that could possibly interfere with the sensor network transmissions.

At this stage, we have only pointed out the reasons for data losses common to the two schemes of data collection. There may be a few data loss factors specific to a particular scheme, but we will elucidate them while analyzing the results for each scheme.

We measured the data losses for varying number of motes and different spatial arrangements in conjunction with the two data collection techniques (autosend and query schemes). The experiments for the last two parameters - distance from the base station

and interference from other devices, were then performed only with the best data collection scheme.

CHAPTER 6

Experimental Results and Analysis

In this chapter we present the experimental results of our spectral camera calibration system. We present the results of thermal IR camera calibration using the temperature sensors on the MICA motes. Calibration results of luminance sensors on the MICA motes using the hyperspectral camera are also shown. One of the system's goal described earlier was minimizing wireless losses during sensor data collection. In this chapter, we first find the optimal sensor network setup by evaluating and analyzing losses under various network configuration parameters. Once the best network setup is determined, it is used in the camera calibration system to gather relevant information from the deployed sensors. The results of the sensor and camera calibration are then described.

6.1 Minimizing Information Loss Via Optimal Network Design

As discussed in Chapter 5, we want to find a sensor network setting that ensures minimum data loss during wireless transmissions. We do so by deploying the network, under different parameters (data collection schemes, node layout, number of nodes, distance form base station and interference from other devices), noting the network performance in each setup. We will first analyze the two data collection schemes individually with the three sensor node layouts we mentioned earlier. In each experiment, we start with one node and go up to seven nodes. The distance and interference experiments are only conducted with the best data collection scheme. It must be noted

that all sensor nodes have similar battery power. All nodes were provided with fresh batteries at the start of the experiment. This fact is important to mention, as battery power will finally determine the strength of data transmissions and receptions. The results of this study on optimal sensor network design were published in [44]. The work presented here is an extension of the NCSA technical report in [45].

6.1.2 Data Collection Schemes

We consider two possible ways to collect sensor data from the network – the 'autosend' and the 'query' scheme.

6.1.2.1 'Autosend' Scheme

The graph in Figure 6.1 shows the percentage of readings lost in 'autosend' scheme as a function of increasing number of motes for each of the three mote arrangements.

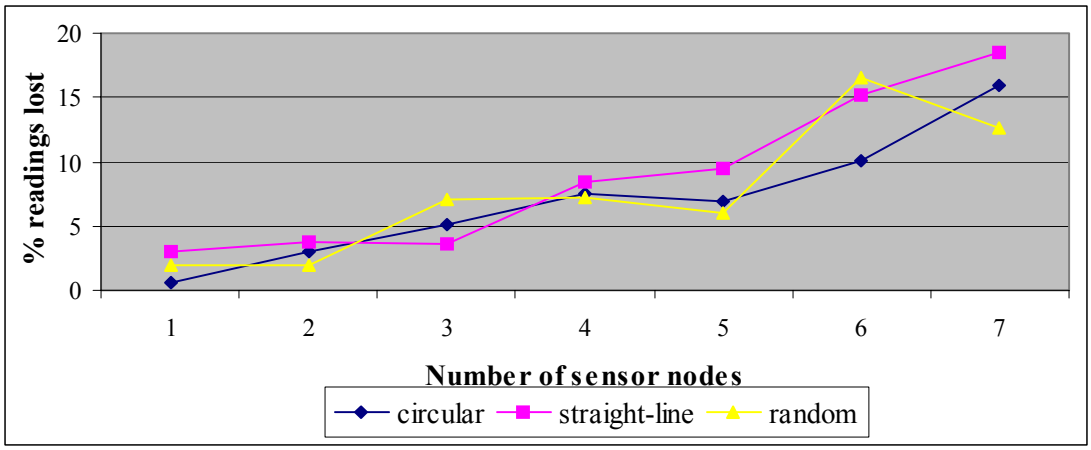


Figure 6.1 Percentage of readings lost in 'autosend' scheme.

From Figure 6.1, we note that in all the sensor network layouts, in general, the percentage of readings lost increases as the number of motes increases. The increase in the percentage of readings lost is gradual in the beginning but spikes drastically as we go on increasing the number of motes. This is because, the number of transmissions increases linearly with the number of motes. When the number of transmissions goes up, so does the probability of a collision and therefore the percentage of readings lost. In some cases, we notice that the percentage of readings lost decreases by up to 4% as we increase the number of motes. We think that this could be due to the random nature of the causes behind lost readings. Wireless telephones and other devices in the laboratory that use the same radio frequency of 916 MHz could also be the reason for this randomness. We noticed that when a phone was in use, almost no packet was successfully transmitted or received by the base station, irrespective of the number of sensor nodes in the network. As far as the mote layout is concerned, the circular arrangement seems to give minimum data losses in almost all cases. The random arrangement gives low data losses in some cases, but in other cases gives the highest losses. One might wonder why the data loss in 'autosend' scheme is not 100% since nodes are synchronized in the beginning and will send a packet with 10 readings simultaneously, every second. This could be credited to the medium access control (MAC) protocol, which is implemented in the lower levels of the TinyOS and is responsible for transmissions. It could also be due to the loss of sync between the sensor nodes (described earlier).

6.1.2.2 'Query' Scheme

Apart from the 'query' scheme described in Chapter 5, we tried other modifications of the 'query' scheme. One of them was to allow a node to send only when the base station queries it and drop the query if there are less than 10 readings on the node. This would reduce the collisions that result when a node sends data out of turn. We tested this method. Since we did not get promising results from this modification, we have not included the exact results of those experiments here. It will suffice to mention that the scheme did not guarantee lower readings losses because it needs perfect synchronization between a node and the base station. If a base station queries a node when the node does not have enough readings, the node will not be able send its data until it is queried again. There is yet another option whereby we send as many readings as are available on a mote when it is queried. This too has its flaw that there will be wastage of network bandwidth and a large number of transmissions will have packets less than half full. After trying these variations of 'query' scheme, we decided to stick to the one described in Chapter 5. The base station waited for about 300 milliseconds after sending a query and before querying the next node.

Figure 6.2. plots the data losses in the 'query' data collection scheme. We see the same trend in data loss as we saw in the 'autosend' scheme. The losses increase as the number of nodes in the network increase. However, a closer look at the numbers show that the percentage of loss is much greater in the 'query' scheme. In the 'query' scheme, apart from readings lost due to collisions, there is another latent reason that could be causing the large number of reading losses.

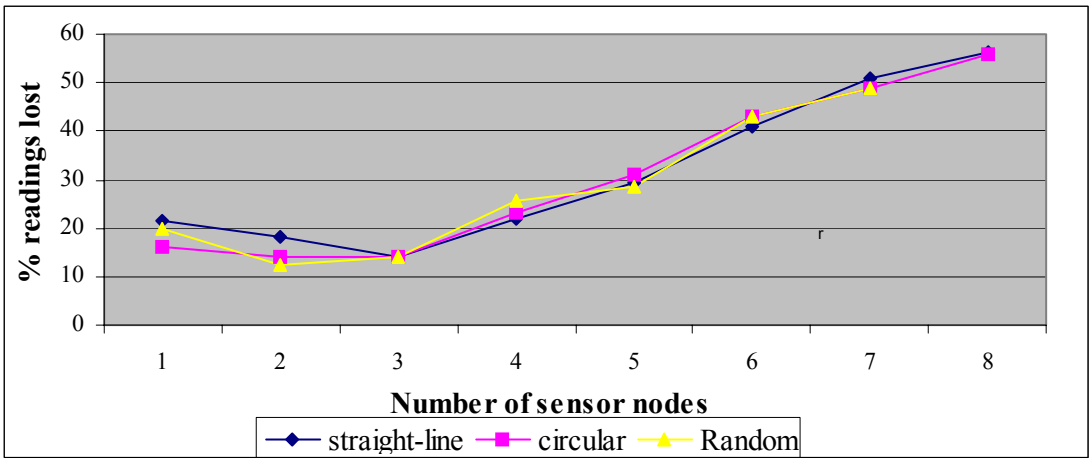


Figure 6.2 Percentage of readings lost in 'query' scheme.

Imagine a situation in the 'query' scheme where a node has a packet with the required number of readings ready, but has not being queried by the base station. Since we have limited the amount of data stored in the local array on a node to 10 readings only, in the next timer interval, even though there is a reading to be stored, it is dropped. Readings are dropped until this mote is queried and the local array is set to blank again. The more the number of motes, the longer a particular mote will have to wait before it is queried, and thus more and more readings will be dropped. This would be a problem in any 'query' scheme operating in a continuous sensing application, unless we remove the limitation on the amount of data stored on a mote.

Another problem occurs if the base station queries a node when it has just started filling its local array. This node will not have 10 readings in the 300 milliseconds for which the base station waits, before it queries another mote. In this case, this node will send the packet later, when is has 10 readings, increasing a possibility of collision with another mote's transmission.

In the ‘query’ scheme, there does not seem to be a clear winner among the three different mote arrangements. All the arrangements give similar results for the ‘query’ scheme. It appears that the distance from the base station or the layouts do not affect the performance of the ‘query’ scheme as much as other reasons like wait-time before a mote is queried and collisions due to out-of-turn transmissions.

6.1.2.3 ‘Autosend’ vs. ‘Query’ Scheme

Comparing the graphs in Figure 6.1 and Figure 6.2, we notice that the percentage of readings lost in ‘query’ scheme is at least 10% higher than the readings lost in ‘autosend’ scheme. As the number of nodes increases, the losses in ‘query’ scheme shoot to about 50% while those in ‘autosend’ stay around 15%. Thus, it is obvious that in terms of the number of readings lost, ‘autosend’ is far superior to the ‘query’ scheme.

Intuitively, the ‘query’ scheme allows the base station to control data transmissions. The number of readings lost due to collisions should therefore be less in the ‘query’ scheme as compared to ‘autosend’ scheme. However, it seems that the ‘query’ scheme introduces other causes for readings to get lost.

Readings are mostly lost in ‘autosend’ scheme due to collisions or initial startup stabilization. However, in the ‘query’ scheme a large number of readings seem to be getting lost either due to collisions and lack of synchronization between the base station

and the sensor nodes. These ‘query’ scheme problems have been discussed in detail earlier.

Another problem specific to the ‘query’ scheme is if a node dies out in between. Since nodes are queried in a round robin fashion, the transmission slot of about 300 milliseconds for the dead node goes wasted. Not only that, at this time other motes may start loosing readings too. Such a failure has no affect on the ‘autosend’ scheme, but adversely affects the ‘query’ scheme.

The experimental results in this section showed that ‘autosend’ scheme is better than the ‘query’ scheme. Hence, for the next two sets of experiments we used the ‘autosend’ scheme for data collection.

6.1.3 Distance from the Base Station

We measured the data loss as a function of increasing distance from the base station until we reached the maximum range of transmission. From our experiments, we found that the maximum transmission range was around 70 feet. In order to eliminate losses due to collisions with other sensor nodes’ transmissions, in this experiment, we used only one sensor that uses the ‘autosend’ scheme for data collection.

Table 6.1 shows the extent to which data loss is affected by distance. We observe that in general, data losses increase with increasing distances from the base station. At 70 feet, there was 100% data loss. Material in radio-transmission literature supports the fact that

the signal power decreases (losses increase) when distance between transmitter and receiver increases. Literature also suggests that many other problems, like those of *'path-loss'* and *'multi-path fading'*, start featuring with increasing distances.

Table 6.1 Percentage of readings lost as a function of increasing distance.

Distance from base station (in Feet)	Percentage of readings lost
1	2.0696617
5	2.0696617
10	3.5482258
20	1.0494742
40	4.9975259
50	3.0484757
55	5.0474762
60	1.9330504
65	1.1046817
70	100

There could also be random collisions with transmissions from other wireless devices (covered in next section). Due to the inter-play of multiple complex factors influencing losses, we are unable to pinpoint an exact cause for the occasional fluctuations in data losses.

This set of experiments has given us an idea of the data loss we can expect if the sensor node is far from the base station. If an application demands high data reliability and delivery, and still wants to cover a good area through its sensor network, it should consider a multi-hop network where sensors form a chain of *'adequately'* closely placed nodes, passing data to the next node which is closer to the base station, until it finally reaches the base station.

6.1.4 Interference from Other Devices

The MICA motes use a radio frequency of 916 MHz for wireless transmissions. In this experiment we tried to determine the extent of interference by the following devices that use radio transmissions for their operations and are commonly found in an indoor environment:

a) Telephones and wireless video transmitters: Many powerful cordless telephones and wireless video transmitters use frequencies in the 900 MHz range. We tested the performance of our sensor network in the presence of such a telephone (EnGenius SN920) and a video transmitter (accompanying CCTV-900 receivers). The results were disheartening because our sensor network had almost 100% data loss the moment these devices were switched on. These losses could not be reduced even when we tried to increase the distance between the transmitting devices (telephone and camera) and the sensors, or decrease the distance between the base station and our sensors.

b) Wireless LAN (802.11b): 802.11b wireless LANs are commonly deployed in offices. These networks use a frequency of 2.4 GHz and do not interfere with our sensor network.

c) Wireless audio transmitters: we operated our network in the presence of audio-technica's ATW-3110D audio transmitters that use frequencies between 655-680 MHz for transmitting sound information to their corresponding receivers. They did not interference with our network.

d) Other independently existing sensor motes: it is possible that there is another sensor network close to ours, which could interfere with our sensors' transmissions. We simulated such an independent network by placing some sensors in the laboratory, which would broadcast meaningless (with respect to our application) data after every 150 milliseconds. The purpose of this setup was to determine how, similarly powered devices using the same 916 MHz frequency but working independently (asynchronously), could hinder our network performance. Figure 6.3 shows how the data losses in our network increased as the number of sensor motes in the 'other' independent sensor network increased.

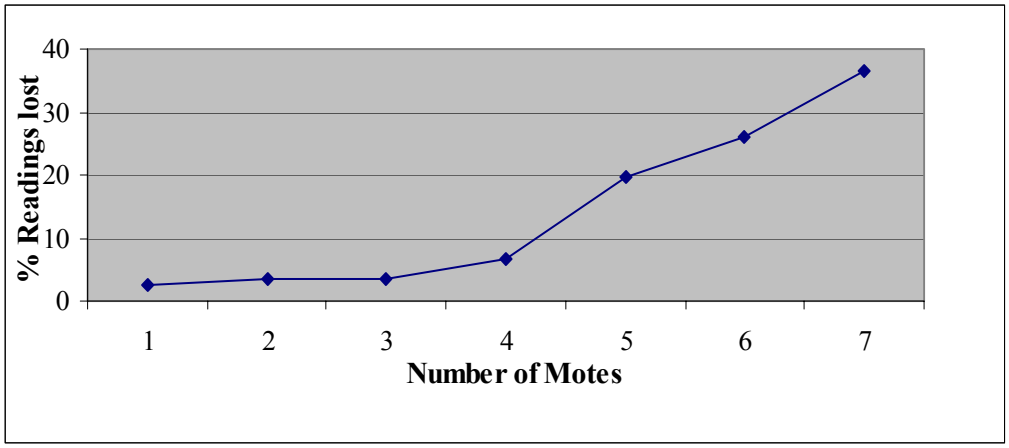


Figure 6.3 Percentage of readings lost as a function of number of motes in the 'other' network.

The graph clearly shows that the data loss is quite significant. The 'other' network is totally out-of-sync with our network and since its nodes are transmitting at a much faster rate than our node in autosend mode, maybe our node's transmissions are suffering more collisions and lesser useful data is being received at the base station.

The experiments in this category have shown that MEMS sensors' transmissions are susceptible to interference from other devices that use the same frequency range for transmissions. We have realized the MICA motes are relatively low powered as compared to bigger devices like wireless cameras and telephones. Thus, in the presence of more powerful devices, a sensor network is prone to total failure. In the presence of similarly powered devices, operating asynchronously with our network, the loss increases linearly as the number of nodes in the 'other' network goes up.

6.2 Calibrating the Thermal IR Camera Using MICA Temperature Sensors

In this section we present the results of calibrating a thermal IR camera. We give the outcome of calibrating the MICA temperature sensors using a thermometer as the spectral gauge. Once we have the calibrated sensors, we show how we obtain the thermal camera calibration. Though we follow the procedure described in Chapter 2 and 5, we also elaborate solutions to some challenges that we encountered while doing the actual calibration. Learning from the results for the optimal sensor networks in previous section, we will use the 'autosend' scheme to collect data from the sensor network.

6.2.1 MICA Temperature Sensor Calibration Results

Once we have collected the MICA raw ADC temperature readings using the 'autosend' data collection scheme, they are converted to engineering units using the manufacturer's (Crossbow Inc.) recommended formula [27]. From our correspondence with Crossbow Inc. representatives, we found that this conversion is only an approximation. Thus, the

temperature sensors must be calibrated by combining the conversion formula with calibration corrections using a thermometer. In our case, according to the manufacturer's data sheet for the thermistor on MTS101CA [27], the raw thermistor reading from the ADC can be converted to degrees Kelvin using the following approximation over the measured temperature range of 0-50 ºC:

$$1/T_{ManufMICA}\,(K) = a + b * Ln(Rthr) + c * [Ln(Rthr)] \wedge 3 \quad \ldots(1)$$

where,

Rthr = R1(ADC_FS - ADC)/ADC,

a = 0.001010024,

b = 0.000242127,

c = 0.000000146,

R1 = 10K,

ADC_FS = 1024,

ADC = raw value from sensor's ADC measurement.

To compute accurate temperature values, the converted raw values based on equation (1) are mapped to the corresponding thermometer reference measurements in °C according to the following equation:

$$T_{CalibMICA}\,(K) = T_{ManufMICA}\,(K) + offset \quad \ldots(2)$$

where,

$$offset = T_{ReferenceThermometer}\,(C) - T_{ManufMICA}\,(K) - 273$$

In order to improve accuracy of readings, we temporally averaged multiple readings from each mote before calibrating it. From our experiments, the manufacturer's formula (1)

gave sensor readings around 31°C. The thermometer measured 21°C. Thus, from the above calibration formula (2), we calculated the offset as 10°C.

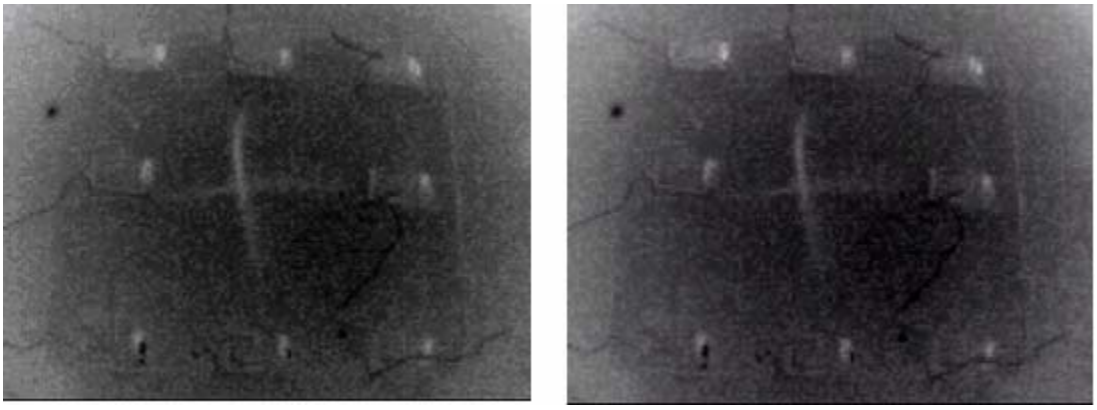
6.2.2 Thermal IR Camera Calibration Results

After calibrating the MICA sensors, we use the calibrated sensors to calibrate the thermal camera (as discussed in Chapter 2 and 5). The thermal IR camera outputs gray-scale images. The Hauppauge WinTV board takes the camera's analog gray scale input and digitizes it into red (R), green (G) and blue (B) values for all image pixels. During the course of our experiments, we realized that due to the camera noise, the RGB values at any pixel in the thermal image are not always identical. We took advantage of the three-band representation of an expected scalar (gray scale) pixel value and increased the stability of the calibrated camera images by temporally averaging image frames until all three RGB values at any pixel converged to almost identical values. The averaged image contained images in increasing order of timestamp. To find an optimal number of images to be used while constructing an averaged thermal image we first defined an allowable difference threshold *'d'* between the R, G and B values of a pixel. Table 6.2 shows the information required to find an optimal number of images used to construct the averaged image. The thermal image contained a total of 230400 pixels. The third column in Table 6.2 gives the percentage of pixels violating *'d'*, for the *'d'* value set to 4. We define *'t'* as the difference between the percentages of violating pixels in two consecutive averaged images. We kept adding more images to the averaged image until *'t'* became about 0.0001. From the table, we see that for 117 images in the averaged image, *'t'* is 0.000191. Thus, we decided to use 117 images to obtain a reliable averaged image.

Table 6.2 Information for determining optimal number of images in averaged image

Number of Images Averaged	Number of Viloating Pixels (d = 4)	% of Violating Pixels (d = 4)	*'t'*
2	32937	0.142956	-
3	30966	0.134401	0.008555
6	21069	0.091445	0.042956
10	14811	0.064284	0.027161
15	10000	0.043403	0.020881
20	8649	0.037539	0.005864
63	4055	0.0176	0.019939
87	3581	0.015543	0.002057
111	3126	0.013568	0.001975
117	3082	0.013377	0.000191

Figure 6.4 shows (a) a single image with (b) a temporally averaged image. In order to improve the robustness and accuracy of our results, we averaged the sensor readings in the same time period as the images used to derive the averaged image. That is, if the 117 images in the averaged image corresponded to the first 15 seconds of the experiment, the first 15 seconds worth of data from each MICA sensor was also averaged.



(a) (b)

Figure 6.4 (a) A single thermal image. (b) Thermal image averaged temporally using 117 images.

The ultimate calibration goal of this work was to determine accurate spectral information at any pixel of a thermal IR camera image at any given time. Since the scene temperature of the thermal camera was not uniform, we had to determine the exact pixel location of the thermistor on each sensor in the thermal image. As mentioned in Chapter 5, it is difficult to see any distinguishing feature in the thermal image. To find the precise location of the thermistor on the sensors, we registered the averaged thermal camera image with a visible spectral camera image of the same experimental setup. We used the i2k [43] software suite's implementation of spatial registration (described in Chapter 5) to register the thermal and visible spectral images.

After computing temporal and possibly spatial statistics, the data from the MICA thermistor sensors and the thermal IR camera are prepared to form a spectral calibration mapping.

Table 6.3 Corresponding values of MICA temperature sensor and thermal IR camera at the MICA sensor locations.

MICA		Thermal IR Image		
ID	Averaged Temperature (°C)	MICA Row	MICA Column	Averaged RGB Value
1	21.52	28	157	104.66
2	22.09	191	151	54.33
3	20.78	416	146	67
4	21.38	417	336	42.33
5	20.99	417	528	47
6	21.66	194	512	80
7	21.41	34	496	80
8	20.94	32	325	72

Table 6.3 shows the calibrated and averaged temperature values recorded by eight MICA temperature sensors and the corresponding averaged RGB values at the pixel locations

(depicted by row and column) of the MICA sensors obtained from the temporally averaged and registered thermal IR image.

We selected a linear spectral calibration function to model the MICA to pixel value mapping. We had expected to observe greater pixel values for higher MICA temperature and smaller pixel values for lower MICA temperature values. However, as is evident from Table 6.3, we noticed inconsistent oscillation of camera pixel values with respect to MICA values. Increasing MICA temperature values did not guarantee increasing thermal image pixel values.

Based on the discussions with the thermal IR camera manufacturer, this problem was associated with inaccuracy of camera spatially varying gain settings. We resolved this issue temporarily by spatially averaging MICA and thermal IR pixel measurements.

We defined the following parameters:

a) Average MICA temperature, μ_{MICA}.

b) Standard deviation in MICA temperature σ_{MICA}.

c) Average thermal IR camera value, μ_{T_IR}.

d) Standard deviation in thermal IR camera value, $\sigma_{T\_IR.}$

The above four parameters were mapped as shown in Figure 6.5.

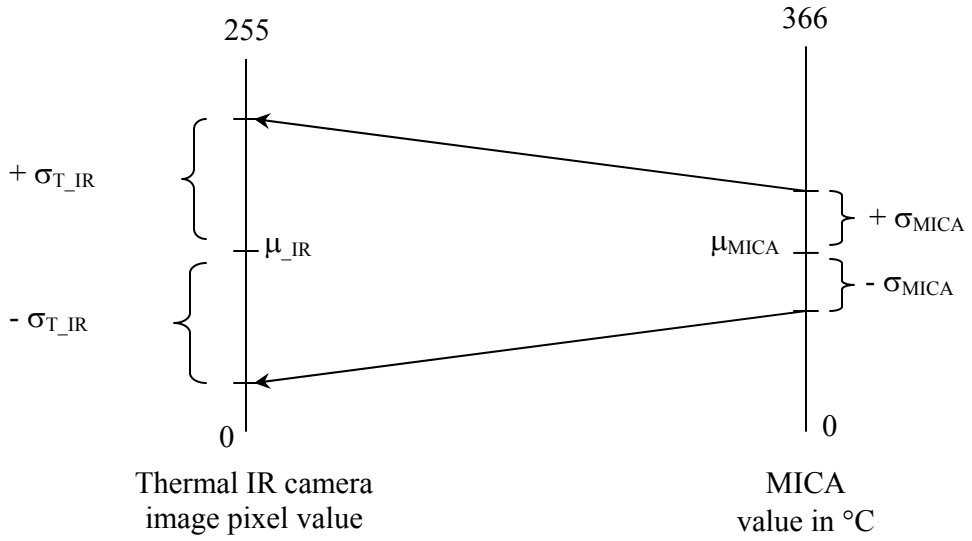


Figure 6.5 Linear mapping between thermal IR values and MICA values

To find the temperature value at any pixel on row 'R' and column 'C' in the thermal image, we have the final linear calibration function:

$$T_{CalibT\_IR}(R, C)\ [°C] = (((PixelValue(R,C) - (\mu_{T\_IR} - \sigma_{T\_IR})) / 2\sigma_{T\_IR}) * 2\sigma_{MICA}) + (\mu_{MICA} - \sigma_{MICA}) \quad \ldots(3)$$

From Table 6.3, we obtained:

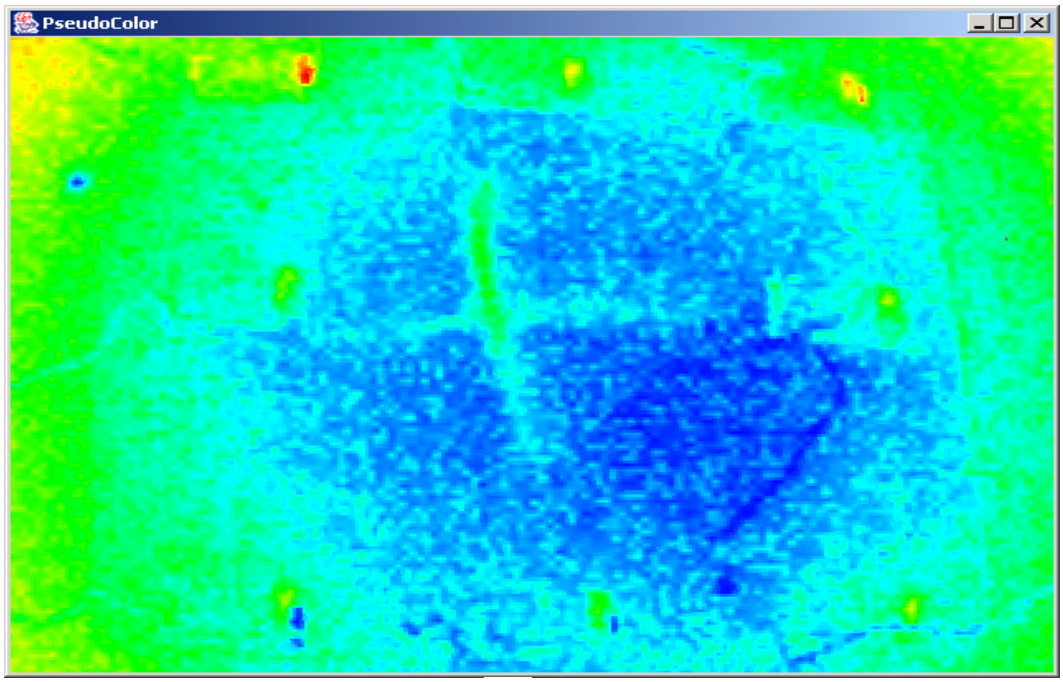
$\mu_{MICA} = 21.34$ °C,

$\sigma_{MICA} = 0.43$ °C,

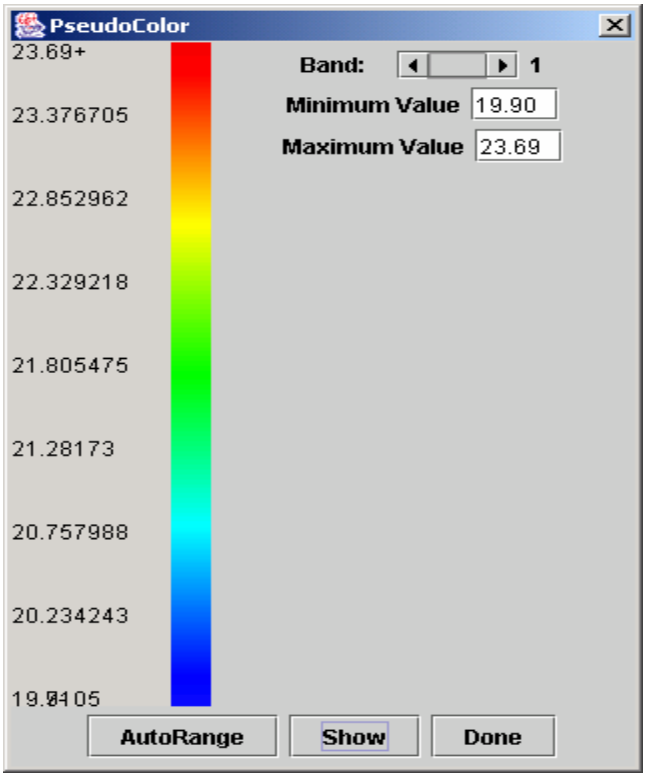
$\mu_{T\_IR} = 68.415$ and

$\sigma_{T\_IR} = 20.474$

The resulting calibrated thermal IR image is shown in Figure 6.6.



(a)



(b)

Figure 6.6 (a) Pseudo-colored calibrated thermal image. (b) Temperature scale for (a)

Figure 6.6 shows that the scene temperature varies between 19.9°C and 23.67°C. The blue color depicts the coolest regions while red the hottest. The red area is a metallic switch that is hotter than any other object in the thermal camera's field of view. The image above, with the temperature value at every pixel gives more information than the grayscale image given by the thermal camera. Thus, the use of calibrated thermal camera in a smart space helps in better inferring the scene and making intelligent decisions.

6.3 Calibrating the MICA Luminance Sensors Using the Hyperspectral Camera

In this section we address the issue of calibrating a luminance sensor on the MICA mote with the hyperspectral camera. This work is still ongoing and the presented results, although incomplete, demonstrate the calibration concept. The purpose of calibrating the luminance sensor is to be able to map a raw value from the luminance sensor to the illumination due to the light source. In a hazard aware space, this conversion of raw sensor values could help in determining if the sensor was exposed to a light of unexpected or abnormal illumination.

We first calibrate the hyperspectral camera and then calibrate the luminance sensor on the MICA mote. The hyperspectral camera used for this experiment was manufactured by Opto-Knowledge Systems Inc. [5]. The experiment consisted of first placing a MICA sensor in the field of view of the hyperspectral camera. The light incident on the sensor was then varied from 0 Watt to 15 Watts, with 9 intervals in between. Continuous sensor readings were collected for the duration of the experiment. Images were taken from the hyperspectral camera for each of the nine light intensities.

6.3.1 Hyperspectral Camera Calibration Results

Figure 6.8 shows the values obtained from the hyperspectral camera for each of the nine different light illumination values plotted in terms of power (in Watts). The hyperspectral data were obtained by averaging the difference between hyperspectral values and camera noise values (measured with the lens cap on) over a sub-area with a 97% reflective background.

From the graph, we notice that for a small increase in power P, there is a large increase in the hyperspectral camera value 'C'. We thus assumed an exponential relationship between the camera values and power and used the following relation between P and C:

$$P = k_1 * Ln(C + k_2) + k_3 \quad \ldots(4)$$

where k_1, k_2, k_3 are constants.

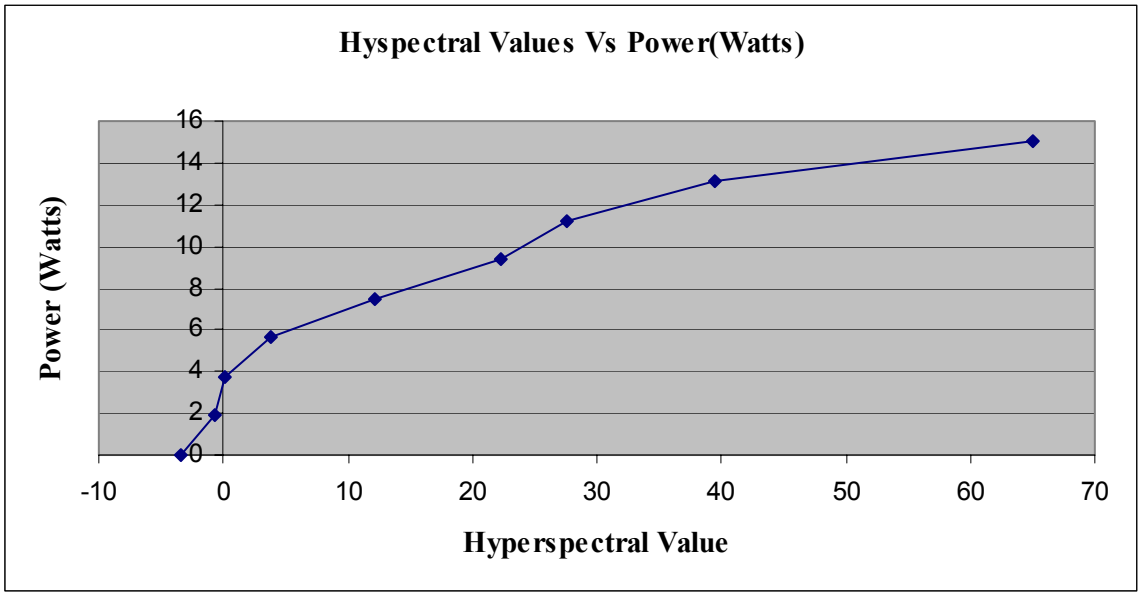


Figure 6.8 Hyperspectral camera values with increasing wattage

The constant k_3 was added to the camera value C to compensate for the noise in the collected data. We observed that even when the illumination was zero, we measured a non-zero, in fact a negative camera value. We added a positive constant k_3 to make the camera value positive.

Solving equation (4) using non-linear regression, we obtain:

$$P = 5.4974 * \text{Ln}(C + 8.2013) - 8.6068 \quad \ldots(5)$$

Equation (5) is used for converting hyperspectral image values to light power magnitude (proportional to illumination).

6.3.2 MICA Luminance Sensor Calibration Results

According to the manufacturer specification of the MICA luminance sensors, the resistance across the sensor and the light illumination has a linear relationship on a logarithmic scale, with the resistance having the highest value when illumination is zero. This type of dependency was experimentally measured and the measured data are plotted

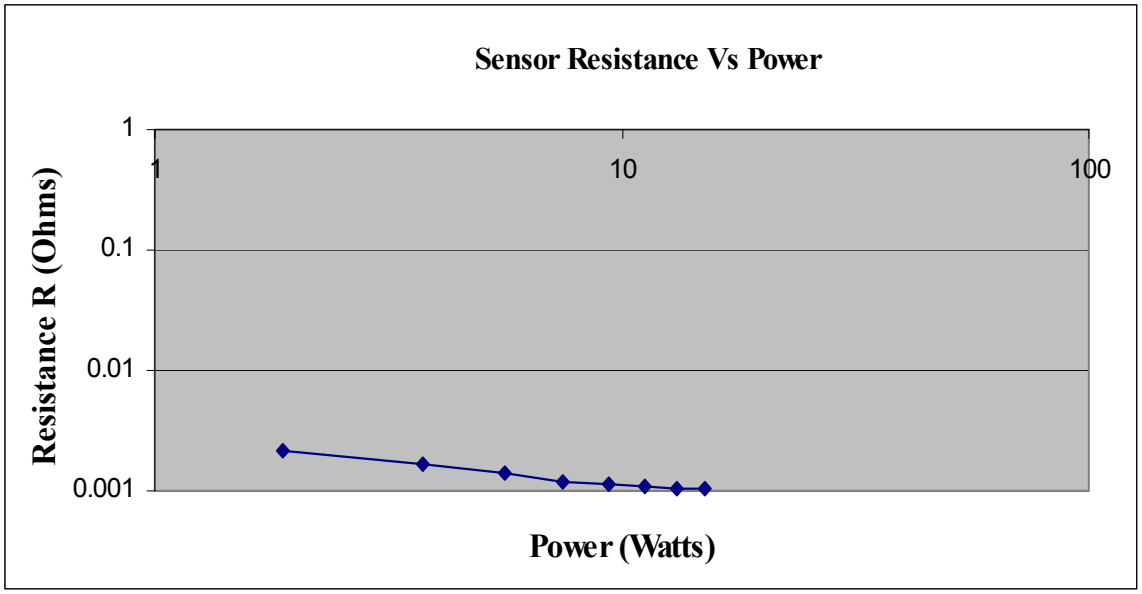


Figure 6.9 Sensor resistance against increasing power values (logarithmic scale).

in Figure 6.9.

Since the power P used by a light source is proportional to the illumination ($P \propto$ Illumination), we express our results in terms of the power P instead of illumination. Thus the relationship between the resistance R and power P will be of the form:

$$\text{Log}(R) = a_1 * \text{Log}(P) + a_2 \quad \ldots(6)$$

The raw value S collected from the sensor is the reciprocal of the resistance, that is,

$$1/S \propto R.$$

Thus equation (6) can be rewritten as:

$$\text{Log}(a_3/S) = a_1 * \text{Log}(P) + a_2 \quad \ldots(7)$$

$$\text{Log}(S) = - a_1 * \text{Log}(P) + \text{Log}(a_3) - a_2 \quad \ldots(8)$$

$$\text{Log}(S) = - a_1 * \text{Log}(P) + a_4 \quad \ldots(9)$$

where $a_4 = \text{Log}(a_3) - a_2$

We have already calibrated the hyperspectral camera in the previous section and have equation (5), which gives us the relationship between P and the camera value C. Thus we can substitute P with camera value C, and solve equation (9) to get the values for a_1 and a_4.

$$\text{Log}(S) = - a1 * \text{Log}(5.4974 * \text{Ln}(C + 8.2013) - 8.6068) + a4 \quad \ldots(10)$$

Using linear regression to solve equation (10) we obtain:

$$\text{Log}(S) = 0.39825* \text{Log}(P) + 2.5479 \quad \ldots(11)$$

In Figure 6.10, we plot known values of S with values of P predicted by equation (11).

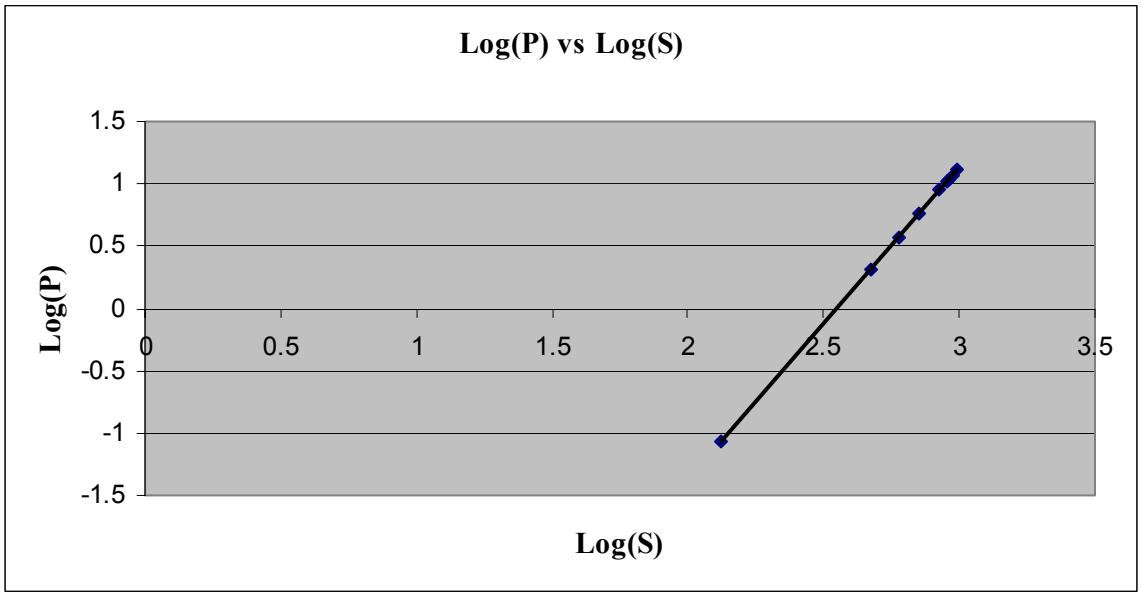


Figure 6.10 Graphical representation of known values of Log(S) with predicted values of Log(P)

As we can see from the graph, there is a linear relation between the logarithmic values of S and P. According to equation (11), one can convert any raw luminance sensor value to a light power magnitude to find the amount of illumination incident on the luminance sensor.

CHAPTER 7

Conclusions

In this work, we have described a novel spectral camera calibration technique using wireless, 'smart' MEMS sensors. Specifically, we have shown the calibration technique for a thermal camera. At the end of the calibration, we were able to derive a linear calibration function that assigns spectral (temperature) values to individual pixels of the thermal image, thereby increasing the information gained from the image. We also derived a calibration transformation between the MICA luminance sensors and the power used by a light source, with the help of a hyperspectral camera. Moreover, the data collection, analysis and subsequent calibration steps of the system require minimal human intervention and can be integrated into a single software suite. These features of our system make it particularly attractive to be used in hazard aware spaces.

We presented the software and hardware specifications of the MICA sensors. The strengths and limitations of the MEMS sensors, as given in the current sensor network literature were discussed. Since energy is the biggest constraint for the MEMS sensors, we tried to design an optimal single hop sensor network that minimized energy loss due to data collisions. We chose a single hop network as opposed to a multi hop network in order to understand the less complicated network. The results obtained in this study establish a lower bound on the amount of data lost in a multi hop network. Our experiments to evaluate different parameters of a sensor network have given helpful insights for an optimal sensor network design. For an application like ours that needs

continuously sensing and transmission of sensor data, the 'autosend' scheme of data collection gave lower data losses. Circular arrangement of nodes around the base station was better than the random or straight-line spatial arrangement. The number of motes in the network determines the amount of traffic that will be generated and thus the losses that would occur due to collisions. Finding the optimal number of motes that will fulfill the application's needs and yet keep the losses within acceptable bounds is important. Moreover, the MICA motes are very sensitive to the presence of other wireless devices that use the same wireless frequency of transmission. While deploying a sensor network, one must be aware of the extent of data losses that an application would incur due to these other devices. The experimental results of our study can serve as guideline in future application design.

Once the optimal sensor network was laid out, we wanted to maximize the information content of the collected data. Various intermediate steps lead to a more stable and accurate calibration system. We synchronized the different data sources and timestamped readings from each source to allow precise comparisions during the analysis stage. Temporal averaging of the sensor readings and camera images was done to eliminate the noise during data collection. Spatial registration between the thermal camera image and the visible spectrum image was also incorporated to get better measurements.

We believe that our camera calibration system with all its above components is the first of its kind. Our method is quite general and future direction of our work is to use a similar approach to calibrate other spectral cameras.

References

[1] ASPRS Camera Calibration Panel Report. Available at http://www.asprs.org/asprs/news/archive/executive_summary.html

[2] Official Crossbow Inc. website. Available at http://www.xbow.com

[3] TinyOS official website. Available at http://webs.cs.berkeley.edu/tos

[4] Omega thermal IR camera. Available at http://indigosystems.com

[5] Official Opto-Knowlegde Systems Inc. website. Available at http://www.techexpo.com/WWW/opto-knowledge/index.html

[6] T. Chrien, C. Chovit and P. Miller. "Imaging Spectrometry Using Liquid Crystal Tunable Filters". Jet Propulsion Laboratory and Cambrigde Research & Instrumentation Inc.

[7] S. C. Lee, S. K. Jung and R. Nevatia, "Integrating Ground and Aerial Views for Urban Site Modeling".

[8] E. Trucco and A. Verri. "Introduction to Techniques for 3-D Computer Vision". Prentice Hall. 1998.

[9] Z. Zhang. "A Flexible New Technique for Camera Calibration". Available at: http://research.microsoft.com/~zhang

[10] B. Caprile and V. Torre. "Using Vanishing Points for Camera Calibration". *The International Journal of Computer Vision*, 4(2), 127-140, March 1990.

[11] G. Stein. "Accurate internal camera calibration using rotation, with analysis of sources of error". In *Proc. Fifth International Conference on Computer Visio*n, pages 230–236, Cambridge, Massachusetts, June 1995.

[12] J. Hartmann, J. Fischer, and P-T Bundesanstalt. “Calibration and investigation of infrared camera systems applying blackbody radiation”. SPIE Proceedings Vol. 4360.

[13] A. Irwin, J. Oleson, Richard Robinson. “MIRAGE: Calibration Radiometry System” Proc. SPIE Vol. 4027, p. 387-398. July 2000

[14] Campbell, J. B., Introduction to Remote Sensing, The Guilford Press, 1996.

[15] Dinguirard, M. and P.N. Slater, 1999; "Calibration of Space-Multispectral Imaging Sensors: A Review"; Remote Sensing of Environment, 68(3): 194-205.

[16] R. Knuteson, B. Whitney, H. Revercomb H., and Best H., "The History of the Atmospheric Emitted Radiance Interferometer (AERI) Prototype During the Period April 1994 through July 1995" - June 1999 (DOE Tech. Memo. ARM TR-001.l). Available at http://www.arm.gov/docs/documents/tech_reports/index.html

[17] Thermal Monitor by Leecorp Limited. Available at: http://www.leecorpweb.com/HumanTempDetector.pdf

[18] ThermaStat by Indigo Systems Inc. and WinSoft Corp. Available at http://www.indigosystems.com/company/PR/pr_030731.html. Also available at http://www.winsoft.com/html/infrared_applications.html

[19] Raghavendra C.S. and Suresh Singh. “PAMAS: Power Aware Multi-Access protocol with signaling for ad hoc networks”. *ACM Computer Communication Review, vol. 28, no. 3, pp. 5—26, July 1998.*

[20] Wei Ye, John Heidemann and Deborah Estrin: “An Energy-Efficient MAC protocol for Wireless Sensor Networks”.

[21] A. Woo and D. Culler, A transmission control scheme for media access in sensor networks, *Proc. 7th ACM/IEEE International Conf. on Mobile Computing and Networking,* Rome, Italy, 2001, 221-235.

[22] L. Kleinrock and R. Tobagi. Packet switching in radio channels, part 1: Carrier sense multiple-access models and their throughput-delay characteristics.

[23] S.S. Lam. " A carrier sense multiple access protocol for local networks." In *Computer Networks, volume 4,* pages 21 – 32, 1980.

[24] Bhaskar Krishnamachari, Deborah Estrin and Stephen Wicker. "Modelling Data-Centric Routing in Wireless Sensor Networks". In Computer Engineering Technical Report CENG 02-14.

[25] C. Intanagonwiwat, R. Govindan and D. Estrin. "Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks," ACM/IEEE International Conference on Mobile Computing and Networks (MobiCom 2000), August 2000, Boston, Massachusetts.

[26] Jeremy Elson and Deborah Estrin. "Time Synchronization for Wireless Sensor Networks". In 2001 International Parallel and Distributed Processing Symposium (IPDPS), 2001.

[27] J. Elson, Lewis Girod and D. Estrin. "Fine-Grained Network Time Synchronization using Reference Broadcasts". Submitted for review, February 2002.

[28] Hauppauge WinTV board. Available at: http://www.hauppauge.com

[29] MICA specification sheet. Available at http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/MICA.pdf

[30] MTS sensor board specification sheet Available at http://www.xbow.com/Support/Support_pdf_files/MTS-MDA_Series_User_Manual_RevB.pdf

[31] MICA: The Commercialization of Microsensor Motes. Available at http://www.sensorsmag.com/articles/0402/40/

[32] TCP Specification. Available at http://www.ibiblio.org/pub/docs/rfc/rfc793.txt

[33] P. Buonnadonna, J. Hill and D. Culler, "Active Message Communication for Tiny Networked Sensors".

[34] H. Wang, D. Estrin and L. Girod, "Preprocessing in a Tiered Sensor Network for Habitat Monitoring".

[35] A. Mainwaing, J. Polastre, R. Szewczyk, D. Culler and J. Anderson, "Wireless Sensor Networks for Habitat Monitoring".

[36] D. Estrin, R. Govindan, J. Heidemann and S. Kumar, "Next Century Challenges: Scalable Coordination in Sensor Networks".

[37] N. Correal and N. Patwari, "Wireless Sensor Networks: Challenges and Opportunities".

[38] P. Gupta and P.R. Kumar, "The capacity of wireless networks". IEEE Trans. On Information Theory, 46(2): 388 – 404, March 2000.

[39] J. Heidemann, F. Silva, C. Intanagonwiwat, R. Govindan, D. Estrin and D. Ganesan, "Building Efficient Wireless Sensor Networks with Low-Level Naming". *18th ACM Symposium in Operating Systems Principles,* October 21-24, 2001.

[40] Java Media Framework. Available at: http://java.sun.com/products/java-media/jmf

[41] Web documentation of MICA code. Available at http://alg.ncsa.uiuc.edu/do/documents

[42] Teillet, P.M., B.L. Markham, J.L. Barker, J.C. Storey, R.R. Irish, and J.C. Seiferth, 2000b; "Landsat Sensor Cross-Calibration Using Nearly-Coincident Matching Scenes"; Proceedings of SPIE Conference 4049, Orlando, Florida, 12 pages, in press.

[43] P. Bajcsy et al. "Image To Knowledge (i2k)" software suite. Available at: http://alg.ncsa.uiuc.edu/do/tools/i2k

[44] S. Saha and P. Bajcsy. "System Design Issues In Single-Hop Wireless Sensor Networks". In Proc. 2nd IASTED International Conf. on Communications, Internet and Information Technology. November 2003. Scottsdale, Arizona.

[45] S. Saha and P. Bajcsy. "System Design Issues for Applications Using Wireless Sensor Networks". Technical Report. Available at http://alg.ncsa.uiuc.edu/documents/TR-20030823-1.doc