Master's Thesis
Ganymed Stanek, ETH D-ITET

November 2004 - April 2005

Energy Conserving Camera Scheduling Policies for Sensor Networks

Advisor UCSC: Professor Roberto Manduchi
Advisor ETH: Matthias Bossardt
Supervisor ETH: Professor Bernhard Plattner

Department of Computer Engineering, UC Santa Cruz
Computer Engineering and Networks Laboratory, ETH Zürich

Contents

List of Tables

List of Figures

Abstract

When trying to monitor an area with a battery-operated sensor network featuring nodes equipped with cameras, it is crucial use as little energy as possible. This can be achieved by turning off cameras whenever possible to improve network lifetime. This thesis shows how to efficiently model the monitored area by so called "virtual doors", and pursues an approach based on a constant frame rate for initial detection of objects combined with wake-up calls between the network nodes. A measure in the form of a miss rate is provided, based on which a decentralized Resource Manager can determine the frame rate and wake-up times for each camera while taking other knowledge into account, such as battery level of the nodes, signals from a PIR sensor, or network congestion. The measures, frame rates and wake-up times are eventually calculated for a real-world scenario with two cameras, and then compared to actual data from the cameras.

Chapter 1

Introduction and Motivation

Scientists, as well as security personnel, are often confronted with the task of monitoring a designated area to acquire knowledge about objects that have passed through that area. Whether this is a person walking in a hallway, a bird returning to its nest to feed its offspring or a car driving around company grounds, this task is still mainly done with video cameras supervised by humans. While the analysis of data from low data rate sensors, such as light barriers, have been automated for decades, the automated analysis of a video stream has been made possible in the last decade with the arrival of readily available cheap computers with sufficient processing power. When one considers the task of monitoring large areas with cameras, many of them spread out and far from power outlets, adequate processing power has to be available even at low energy consumption - leading to the new field of battery operated sensor networks. In such networks, it is crucial to save and balance the use of energy wherever possible to enhance the lifetime of the whole surveillance network.

Energy can be saved by using hardware with low power consumption as well as by using energy efficient protocols for communication inside the surveillance network [RSPS]. In *camera* sensor networks, the energy consumed by sensing and processing is no longer negligible; on the contrary, it may dominate the overall power consumption in a sensor node. Much energy could be saved if the surveillance cameras and their processing computers could be in sleep mode as long as the target scene remains unchanged. The cameras would only turn on from time to time to see if an object has entered their field of view. The cameras in the interior of an area covered by cameras could reduce the rate at which they turn on to detect new objects, if they could rely on being woken up by their neighboring cameras when objects are expected to enter their field of view. Of course, such ways of conserving energy goes hand in hand with a growing number of objects not being detected by the camera network and thus sacrificing quality of service of the surveillance task. In many cases, a small loss in quality might be justifiable if the network lifetime could be improved significantly, which would in turn lead to an actual improvement of the overall quality of service.

This thesis is part of a NASA funded research project, called Meerkats, that tries to combine detection as well as tracking of objects with many cameras with non-overlapping fields of view in a very low power environment by finding an energy/quality trade off (see [BLM$^+$]). The network's given tracking goal is to take at least one snapshot with each camera of each object crossing its field of view. While it is part of the project to save energy wherever possible, the scope of this thesis is to provide a yet to be implemented *Resource Manager* with recommended camera snapshot times, and a *measure* expressing the quality of the time estimations.

The Resource Manager[1] will constantly weigh the *measure* with the remaining battery capacity of the network node, the needed level of quality of service, network congestion, importance of certain regions and other factors to determine the exact time to resume from standby for each camera, communication module or even for each processing unit. If a specific node has the advantage of having more energy available than others, for instance by being connected to a bigger battery, that node can be more generous in allocating snapshots.

The measure provided to the Resource Manager will be called *miss rate* and can be thought of as the percentage of objects that cross a specific camera's FOV without ever being photographed and will be defined formally in chapter 4.

A good indicator of when to turn on a camera could be a signal from a passive infrared sensor (PIR) that detects an approaching object before it has arrived - given that the object emits detectable infra red rays. We will now show reasons why purely PIR based turn-on times are suboptimal and motivate a combination with the calculation based approach pursued in the

[1]The Resource Manager should not be thought of as a single process, but as a group of collaborating processes running on the various sensor network nodes.

rest of the thesis.

First, it's not easily possible to trigger interrupts from external sensors that resume the node from standby on all platforms. Assuming this was possible, one has to be aware that the evaluation of PIR signals can take up to 1 second until accurate information is available according to [ua]. When objects are moving quickly and the fields of view of the cameras are small, the signal from the PIR might not be available until after the object has crossed the camera's FOV.

If many objects were approaching the camera and PIR sensor, there would be many snapshots showing the same objects repeatedly, which could have been photographed with fewer snapshots. Repeated photos of the same objects during the same passage of a camera's FOV are a waste of energy, according the definition of our surveillance task. Due to false positives, which are inherent to PIR sensors (see [GJV+]), only a small amount of objects are needed so that the the PIR sensor constantly triggers. In these cases, snapshots should be taken at a constant frame rate, determined by the average time the objects will spend within the FOV of a camera. Chapter 4 calculates that frame rate which can be interpreted as the lowest frame rate to guarantee a certain miss rate. The higher that frame rate will be, the fewer objects will be missed and vice versa. This frame rate can also be used as a worst-case scenario to benchmark the power consumption in the mentioned high-traffic case.

The constant rate of chapter 4 could also be used to determine periodical intervals after which the computer should resume from standby to read the PIR sensor and then just use the PIR reading to decide whether to eventually turn on the camera.

For optimal results, the information from the PIR sensor would also be handed to the Resource Manager and added as an additional criterion that the Resource Manager takes into account when deciding on the final snapshot times. This would result in an allocation of snapshots that are sometimes calculation based and sometimes based on data from collaborating sensors.

In this thesis, we will just study the calculation based snapshots and distinguish two types:

- Recurring snapshots to detect new objects (discussed in chapter 4)
- Single snapshots that have been scheduled for a specific time when previously detected objects are expected to be be in the field of view (FOV) of the scheduled camera (discussed in chapter 5)

Throughout the thesis, we will use the term *wake-up* for recovering from standby-mode for a snapshot of the second type. In other words, a camera will *wake up* another camera. We refrain from using the term for the self-scheduled, periodic snapshots of the first type. Instead, we speak of periodically *turning on* the camera or *resuming* from standby.

Chapter 2 starts out with a short description of the hardware used in the Meerkats project, which was also the hardware, that was in mind when designing the camera scheduling policies. The chapter will then begin with the contribution of this thesis by introducing the way the environment is being modeled. Chapter 3 discusses related work. Chapter 4 addresses the first type of wake-up with a function assigning a *miss rate* to a constant *frame rate*. Chapter 5 addresses the second type of snapshots by assigning a *miss rate* to a *wake-up time*. In chapter 6, these miss rates for given frame rates and wake-up times are calculated for a real world example and compared to the actual values obtained through a simulation run on data from actual video streams. Chapter 7 concludes the thesis and provides starting points for future exploration of the topic.

Chapter 2

Meerkats Hardware and the Environment Model

The first section in this chapter gives an overview of the hardware in mind when designing the energy scheduling policies. The remaining sections cover the mathematical models for the surveillance environment consisting of cameras and so called *virtual doors*, on which the calculations in chapter 4 and 5 will be based.

2.1 Hardware of the Meerkats Project

In the UCSC Meerkats project, the sensor network will be built of nodes using Crossbow Stargate computers running embedded Linux on Intel X-Scale processors. These boards are equipped with PCMCIA wireless LAN cards and Logitech Quickcam Pro 4000 webcams (similar hardware as used for the Panoptes camera sensor network node, see [FCK$^+$]). The webcam and the wireless LAN card can individually be turned on and off, while the processor features a sleep function which sends it to a standby mode for a specified time. Table 2.1 gives an overview of the various states the Stargates can be in and how much power is consumed in these states.[1] (see: [BLM$^+$])

[1]In order to put the processor into sleep mode, one must execute the utility sys suspend, specifying the time the processor should be sleeping. To put the wireless card in sleep mode, the utility to be used is cardctl suspend, while cardctl resume changes the wireless card from sleep to idle. The mechanism we use to put the webcam in sleep mode is to remove the corresponding modules from the kernel (rmmod usbohci- sa1111), while to change the webcam from sleep to idle, we insert the corresponding modules (insmod usb-ohci-sa1111).

State	Processor	Sensor	Radio	Storage	Power(W)
Sleep	sleep	sleep	sleep	sleep	0.28
P-idle	idle	sleep	sleep	idle	0.56
P-active	active	sleep	sleep	idle	1.60
PR-idle	idle	sleep	idle	idle	1.27
PR-active	active	sleep	idle	idle	2.35
PR-rx	idle	sleep	active	idle	2.29
PR-tx	idle	sleep	active	idle	2.48
PRS-idle	idle	idle	idle	idle	2.00
PRS-active	active	idle	idle	idle	3.09
PRS-sens	idle	active	idle	idle	2.23
PRS-rx	idle	idle	rx	idle	2.94
PRS-tx	idle	idle	tx	idle	3.11
PS-idle	idle	idle	sleep	idle	1.28
PS-active	active	idle	sleep	idle	2.35
PS-sens	idle	active	sleep	idle	1.51
PT-read	idle	sleep	sleep	read	1.34
PT-write	idle	sleep	sleep	write	1.37
PRT-read	idle	sleep	idle	read	2.11
PRT-write	idle	sleep	idle	write	2.14
PST-read	idle	idle	sleep	read	2.10
PST-write	idle	idle	sleep	write	2.14
PRST-read	idle	idle	idle	read	2.81
PRST-write	idle	idle	idle	write	2.84

Table 2.1: Stargate energy consumption characterization ([BLM+]-Cintia Margi)

The boards do not offer the possibility of being woken up from standby mode by the communication core, but it would be possible to combine the Stargate boards with Crossbow Motes [RSH], which are less powerful embedded computers that consume less energy and offer low data rate communication interfaces. These Motes could then wake up the Stargate board via a reserved interrupt.

As long as the Stargate boards are not combined with additional hardware, the decision of when to resume from standby mode has to be evaluated before going into standby. Additionally, when we want to be able to send messages between the nodes (i.e. for informing other nodes of approaching

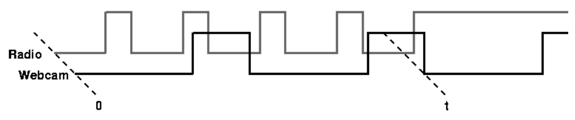


Figure 2.1: At time t, camera sees an object of interest. During the next radio period, node sends wake-up signals to its neighbors and transmits the image to the sink ([BLM+]-Jay Boyce)

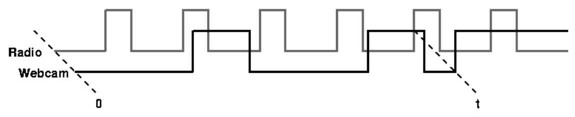


Figure 2.2: At time t, the radio receives a camera wake-up message from a neighbor. The Stargate turns on the webcam and begins taking pictures. ([BLM+]-Jay Boyce)

objects) all nodes have to share a certain time span when they are all awake to interchange messages. On the other hand, the nodes also have to turn on periodically to take snapshots in order to detect new objects entering the monitored area. While this snapshot frame rate for the initial detection of objects varies from camera to camera, the interval in which nodes interchange messages has to be kept strictly synchronous between nodes by a synchronized clock. Figure 2.1 shows how a webcam periodically takes snapshots and detects an object at time t. In the next radio interval it sends a wake-up message to the other nodes and starts transferring the image of the object to the sink. The wake-up message will be received by another camera as shown in Figure 2.2. The wake-up is a message containing the expected arrival time of the object. The receiving node then considers this time when setting its next sleep interval to schedule an additional time to turn on beside the ones already scheduled due to its own detection frame rate.

Every node maintains timers for when to resume each device (e.g., camera, wireless card). If these timers are large enough, the Stargate will begin a timed sleep for the entire node to minimize energy usage. At any given point in time, a node will load only the modules corresponding to the peripherals to be activated. However, the discrete influence of the radio synchronization interval might lead to cases where the node can't be turned on at the time calculated. This influence will be covered in section 5.3.

2.2 Model of the Cameras

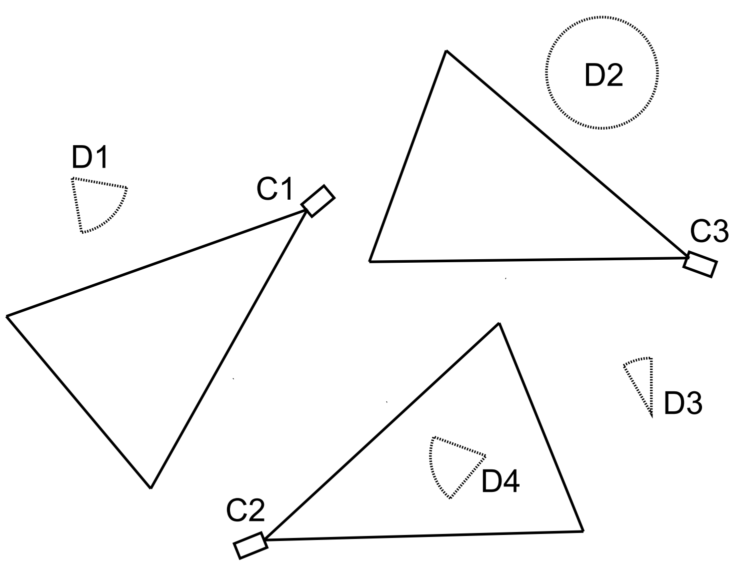


Figure 2.3: Camera Placement and Modeling Flow of Objects with *Virtual Doors* D1-D4

The layout of the cameras in the monitored area is obviously very important for the performance of the overall system. [MD] and [ES] have studied the problem of visibility analysis and optimal placement. In this work we consider a sparse placement of cameras (as in [MM], [KJRS]) to maximize the covered area. This means that, in practice, the FOV of two cameras will seldom overlap.

Figure 2.3 depicts a top down view of a setup of 3 cameras in an open space. We represent a camera's FOV as a triangle,[2] which approximates the actual FOV assuming that the camera is not placed too high off the ground. If the cameras are placed high (e.g. on the ceiling) pointing down, then the ground area visible in the camera is of the form of a trapezoid, leaving an uncovered area between the camera and the trapezoid. Depending on the height of objects in comparison to the camera elevation, the objects will still be partly covered by the cameras when moving from the trapezoid towards the camera - making the perceived FOV again more triangular. The camera's FOV length is limited by the resolution of the camera, if not also by some obstruction or by the fact that the camera is pointing down. As long as the cameras don't cover very large angular ranges, or are omnidirectional, a triangle is a reasonable approximation even when the FOV is obstructed by walls or trees. Figure 2.4 shows the whole set of parameters used to model one camera.

2.3 Model of Moving Objects and *Virtual Doors*

Deciding on a model for moving objects, such as walking or driving people or animals, is less straight forward. People and animals certainly change their direction of movement frequently in the course of a long time span, like a day, but when reducing the time span as well as the area under supervision significantly, people tend to walk along straight lines to pursue an objective. One can think of people walking in a hallway, along a road or straight to a car in a big parking lot – to name a few examples that can be modeled well by straight lines. In fact, even for a car in a wide curved section of a road, a straight line is a good approximation for its movement.

The straight lines need to have starting points, which we will call *virtual doors*. These can of obviously be physical doors through which people appear, but can also be any other place where objects begin their straight line paths. *Virtual* thus refers to the more general concept of various straight paths originating at a single point. When looking at a real world example of an area under surveillance, one realizes that it doesn't need that many virtual doors and straight lines to approximate the most commonly taken

[2]The formulas in chapter 4 and 5 just contain the object's crossing distance of a FOV for a given walking path and can therefore be calculated for any shape of FOV. The experiments and simulations in chapter 6 although are based on triangular fields of view.

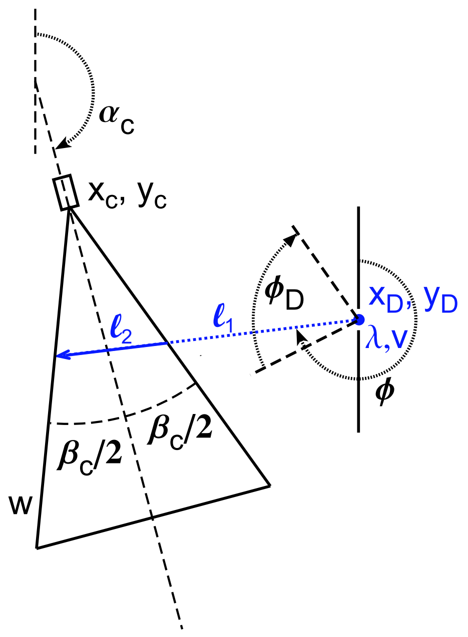


Figure 2.4: camera and door coordinates

walking paths. Indoors, people are often limited in their choices of movement by hallway walls, and therefore are often compelled to walk along straight lines and originate in a few places like physical doors or beginning of hallways. But even outdoors, on an empty field or square, people tend to originate from a few virtual doors that can range from a tree providing shadow to an ice cream stand or a road intersection. Depending what the virtual door in reality refers to, the straight lines originating there are limited to certain orientation angles. Using the example of the tree, the orientation angles could be 360 degrees out of 360 degrees, while a virtual door on a traffic lane probably would emit objects whose orientation angles vary just a couple of degrees as cars are not likely to leave the road. The right hand side of figure 2.4 includes a virtual door with one object crossing the FOV of the camera. Appendix B recommends the vector-parameter-form to model the various straight lines of the walking paths as well as the FOV triangle outlines and gives the formulas for their intersection points.

2.3.1 Object Emission Times at a Virtual Door

For a single virtual door, we expect that the average number of objects emitted in each time unit is equal to a constant λ. The object emission times at each virtual door are then draws from a Poisson process with that parameter λ. The probability that k objects were actually emitted during the time interval T is

$$p_\lambda(k,T) = e^{-\lambda T}\frac{(\lambda T)^k}{k!} \tag{2.1}$$

2.3.2 Speed of an Emitted Object

While different objects, especially when coming from different virtual doors, can differ a lot in the speed in which they move, the actual speed of a single object doesn't change much while it moves on a straight line and is assumed to be constant. The knowledge about the speed of all objects from a single virtual door will be expressed as a truncated Gaussian distribution (2.2) with mean μ_v and standard deviation σ_v. We define the speeds to be positive and express objects moving in the opposite direction by an angle 180 degrees apart instead of a negative speed. We thus truncate the $v \in \ ]-\infty;0[$ part of the Gaussian and scale it so that the total probability remains 1.

$$p_v(v) = \frac{1}{\sqrt{2\pi}\sigma_v}e^{-\frac{(v-\mu_v)^2}{2\sigma_v^2}} \cdot \frac{1}{1-\int_{-\infty}^{0}\frac{1}{\sqrt{2\pi}\sigma_v}e^{-\frac{(v-\mu_v)^2}{2\sigma_v^2}}dv} \tag{2.2}$$

Using the error function from appendix A.1, we can shorten this to

$$p_v(v) = \frac{1}{\sqrt{2\pi}\sigma_v}e^{-\frac{(v-\mu_v)^2}{2\sigma_v^2}} \cdot \frac{1}{1-\frac{1}{2}(erf(\frac{-\mu_v}{\sqrt{2}\sigma_v})+1)} \tag{2.3}$$

$$\int_0^\infty p_v(v)dv = 1 \tag{2.4}$$

This distribution will be used in chapter 4 for objects coming from virtual doors as well as in chapter 5 for the speed of detected objects that are assumed to cross the field of view of another camera.

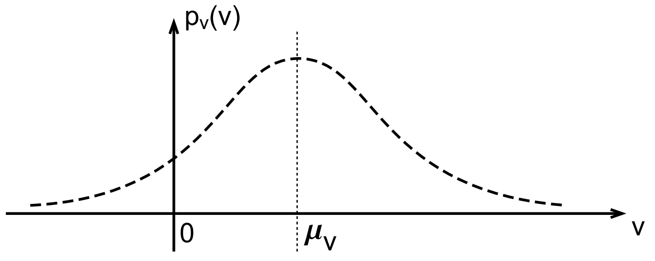


Figure 2.5: Truncated gaussian for the probability distribution of the speed

2.3.3 Angular Range of all Objects Leaving the Door

We assume that all paths originating from a virtual door are contained within an angular sector of angle Φ_D (Fig. 2.4). In the next two subsections, we provide two different distributions that are suitable for modeling object emission angles ϕ.

Uniform Distribution

The uniform distribution is well suited when there is very little to no information about the emission angles at all, which is expressed as large angular sectors ϕ_D close to 360 degrees. In that case, the uniform distribution delivers similar results to a Gaussian distribution with a large uncertainty; but with less mathematical overhead, $p_\phi(\phi)$ will be a constant. The uniform distribution is also a good choice for smaller angles when the distribution really *is* close to uniform.

$$p_\phi(\phi) = \frac{1}{\phi_D} \tag{2.5}$$

Double Truncated Gaussian Distribution

For more advanced modeling of the probability of walking angles, we could also use a truncated gaussian distribution. While the distribution for speeds was solely truncated on one end in section 2.3.2, the distribution for the angles has to be truncated on both ends and then scaled so that the integral over the interval $[0; 2\pi]$ remains one. We use appendix A.1 again to express the integral over the normal curve using the error function.

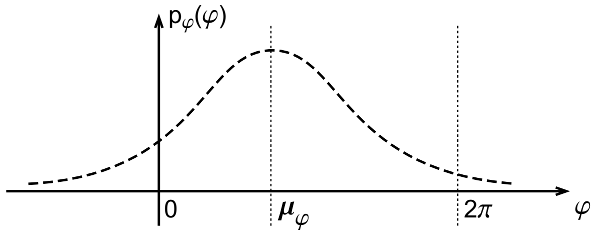


Figure 2.6: Truncated gaussian for the probability distribution of the apex angle

$$p_\phi(\phi) = \frac{1}{\sqrt{2\pi}\sigma_\phi} e^{-\frac{(\phi-\mu_\phi)^2}{2\sigma_\phi^2}} \cdot \frac{1}{\frac{1}{2}[erf(\frac{2\pi-\mu_\phi}{\sqrt{2}\sigma_\phi}) - erf(\frac{0-\mu_\phi}{\sqrt{2}\sigma_\phi})]} \tag{2.6}$$

Chapter 3

Related Work

The following brief report overviews in this chapter are presented in the form of extensions to this thesis instead of alternatives, because there has not been any work published so far that tries to capture objects with a single snapshot or a small number of snapshots while considering energy consumption. The mentioned papers in this chapter complement references to other research throughout this thesis.

[YHS] addressed energy saving in general sensor networks that have overlapping sensing areas. Besides turning off sensors to save energy at locations that are covered by multiple sensors, [YHS] also provides a way to vary the robustness of the whole sensing task between areas by using the signals of several or just single sensors, which is referred to as differentiated surveillance. In contrast to our objective of surveillance where a single snapshot of a each moving object is regarded as complete coverage, the cited paper strives to cover each point of the monitored area at least with one sensor for 100% of the time. This paper could therefore be combined with our work in a remarkable way. Our approach addresses camera layouts with non-overlapping fields of view. To include overlapping field of view cameras, we could use [YHS] to combine several overlapping cameras to create something we could call a *virtual camera* with a FOV that is the combined FOV of all the overlapping cameras. Our algorithm would then be used to decide when to turn on which *virtual camera* and [YHS] would be used to decide which of the cameras within the *virtual camera* eventually should be turned on.

When a surveillance task requires multiple cameras, objects often look very different among the snapshots due to the various artificial and natural lighting conditions the cameras are surrounded by. [PD] uses color calibration between the cameras and stores the video streams not by camera, but by object.

After a snapshot has been taken, it should be compressed before sent it to any further node. [WA] shows that, contrary to popular belief, maximum compression before transmission does not always provide minimal energy consumption, especially in the case of dense sensor networks with complex signal processing algorithms. An algorithm is proposed that selects the optimal image compression parameters to minimize total energy dissipation given the network conditions and image quality constraints. [MLN] comes up with an energy dissipation model, which could also help the Resource Manager to decide on how much local preprocessing should be done before transmitting information.

If the tracking objective is met by only sending the detected objects location to a central node and not the snapshots taken, localized prediction [XL] could reduce the energy consumed even further after applying the approach of chapters 4 and 5. The central node, as well as each other node, predict the path of the objects taken and send messages to the central node when the taken path deviated from what was predicted - reducing the number of long distance transmissions.

The assumption of objects following straight lines between non-overlapping FOV's is also done in [RDD] to calculate the camera calibration parameters (location and orientation of the camera, as well as angular range and resolution). Not addressing energy awareness, many snapshots are taken of each object in order to record *when* and *where* the same objects appear in different cameras, and with that information the calibration parameters can be calculated.

In [DB], objects are tracked between non-overlapping fields of view of uncalibrated cameras by a learning Markov Model. This approach is based on manually identifying the same objects in several cameras to teach the algorithm the probabilities, with which objects move from one camera to another. Knowing this probability helps to identify the redetected objects, but does not contain any information about when the object is going to appear in the other cameras. One could imagine though, that the appearance times would be learned as well. While our approach can be applied in any new environment by precalculating frame rates and wake-up times from the geographical knowledge of the *virtual doors* and camera positions, [DB] requires a learning phase.

Chapter 4

Frame Rate for Initial Detection

We refer to an object slipping through the field of view without being photographed as a *miss*. The higher the frame rate at which the nodes turn on to take snapshots to detect new objects, the smaller the probability of a *miss*. It is therefore very application dependent as to what is regarded as the optimal frame rate. An infinite frame rate would certainly lead to the lowest probability of a *miss*, but is technically not feasible and high frame rates consume too much power in an energy constrained network of cameras. Assuming we would know the rate at which a *miss* occurs for a given frame rate, the former rate could be included in a cost function together with available energy at a node and other factors. The Resource Manager could then find the optimal frame rate by minimizing that cost function. We begin this chapter with the definition of this *miss rate* and a few events to express it formally. The next sections then calculate the *miss rates* associated to the frame rates in a single and multidoor environment.

Let's denote the presence of a moving body originating from door d in the area under surveillance as the event X^1_d. If the body enters the i–th camera FOV (FOV$_i$), we will say that the event $F^1_{i,d}$ occurred. Every time a body from door d circulating in the area covered by the network enters the i–th camera's FOV and is not detected, we will say that a *miss* event for camera i occurred, denoted by $M^1_{i,d}$. A *miss* then does not refer to an empty snapshot, but to an object of which a snapshot has never been taken of. Note, that according to this definition, object B in figure 4 is never considered a *miss*, while object A could lead to a *miss* when no snapshot was taken while it was in the FOV.

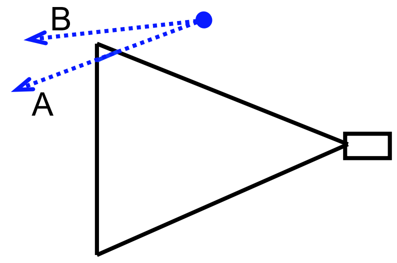


Figure 4.1: About the definition of a "*miss* for A and B"

More in general, one may consider the case of n bodies from door d circulating in the network (event X_d^n), r of which enter the FOV$_i$ at some point (event $F_{i,d}^r$), with the i–the camera missing k of the body in its FOV (event $M_{i,d}^k$). $M_{i,d}^k$ is independent of X_d^n given $F_{i,d}^r$ since objects outside the camera's FOV cannot be detected anyway:

$$P(M_{i,d}^k|F_{i,d}^r, X_d^n) = P(M_{i,d}^k|F_{i,d}^r) \tag{4.1}$$

We will further assume that $P(M_{i,d}^k|F_{i,d}^r)$ is binomial. In other words, each miss event is independent from each other so that there is an equal chance for each miss to happen no matter how many already happened. This makes sense in the case of "rare events", that is, when two bodies are unlikely to appear at the same time in the same FOV. We will also postulate that $P(F_{i,d}^r|X_d^n)$ is binomial, a reasonable assumption in the case of independently moving bodies.

Using these events, we now define the *miss rate*. This *miss rate* will be a measure of performance of a camera node. We define it as the ratio of the expected numbers of *miss* events to the expected number of bodies from door d in the network (*miss rate* or MR$_{i,d}$):

$$\text{MR}_{i,d} = \frac{\text{E}[M_{i,d}]}{\text{E}[X_d]} \tag{4.2}$$

where E[$\cdot$] represents the expectation operator. Let $\text{P}_{M|F} = \text{P}(M_{i,d}^1|F_{i,d}^1)$ and $\text{P}_{F|X} = \text{P}(F_{i,d}^1|X_d^1)$. Using the total probability theorem, and remembering that the conditional distributions of interest are binomial, we can write:

$$\begin{aligned}
\mathrm{E}[M_{i,d}] &= \sum_k k\mathrm{P}(M_{i,d}^k) && (4.3)\\
&= \sum_n \sum_r \sum_k \mathrm{P}(M_{1,d}^k | F_{i,d}^r)\mathrm{P}(F_{i,d}^r | X_d^n)\mathrm{P}(X_d^n)\\
&= \sum_n \sum_r \mathrm{E}[M | F_{i,d}^r]\mathrm{P}(F_{i,d}^r | X_d^n)\mathrm{P}(X_d^n)\\
&= \sum_n \sum_r r\mathrm{P}_{M|F}\mathrm{P}(F_{i,d}^r | X_d^n)\mathrm{P}(X_d^n) =\\
&= \mathrm{P}_{M|F} \sum_n \mathrm{E}[F | X_d^n]\mathrm{P}(X_d^n) = \mathrm{P}_{M|F}\mathrm{P}_{F|X}\mathrm{E}[X_d]
\end{aligned}$$

Hence, from 4.2 and 4.3, we maintain that:

$$\mathrm{MR}_{i,d} = \mathrm{P}_{M|F}\mathrm{P}_{F|X} \tag{4.4}$$

We will now calculate the right hand side factors of (4.4) for objects from door d crossing the FOV$_i$ when snapshots are taken at a constant frame rate for the initial detection of the objects.

4.1 Miss Rate in a Single Door Environment

Fig. 2.4 shows a scenario with a single camera place near a door or an area of relatively high flow of objects. It is likely that this camera will be the first node that can detect a person walking through that door (assuming that there would be other cameras farther away). We modeled the times the persons walk through the door by a Poisson point process of unknown density λ as given in (2.1).

According to our model, persons walk through the door in a rectilinear motion, with constant, but unknown velocity v and orientation ϕ that can be modeled by suitable probability distributions $p_v(v)$ and $p_\phi(\phi)$ as given in chapter 2. Note that prior information on the velocity is often available (e.g. the average speed of walking). We further assume that the orientation and the velocity of motion are statistically independent. As shown in Fig. 2.4, the direction of motion ϕ determines the length $l_1(\phi)$ of the path from the door to FOV$_i$, and the length $l_2(\phi)$ of the path overlapping FOV$_i$.

Together with the velocity v, these path lengths determine the amount of time $t_1(\phi, v) = l_1(\phi)/v$ that it takes to go from the door to FOV$_i$, and the amount of time $t_2(\phi, v) = l_2(\phi)/v$ the moving person will be within FOV$_i$.

Let Φ be the set of orientations that overlap FOV$_i$ as shown in figure 4.2.

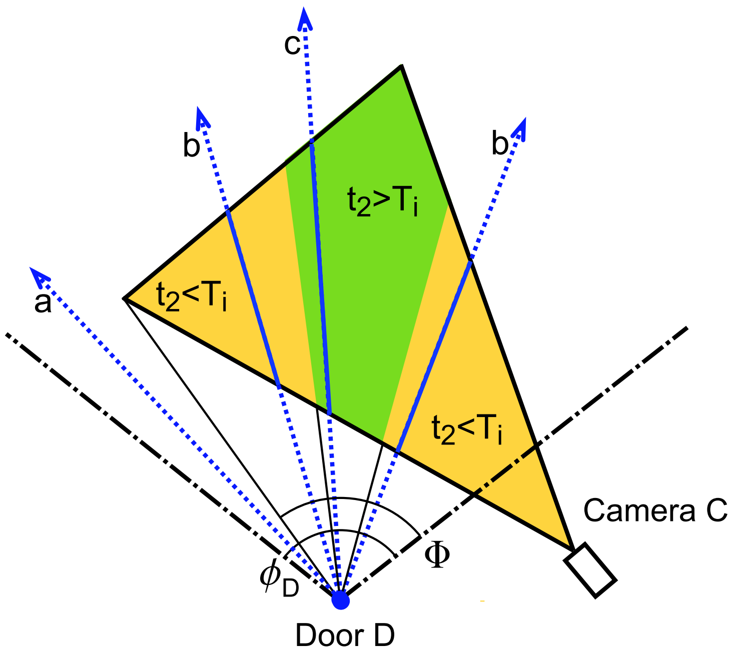


Figure 4.2: Areas and angles of interest when an object originates from a door and crosses the field of view of a camera at a given speed.

Then:

$$\mathrm{P}_{F|X} = \int_{\phi \in \Phi} p_\phi(\phi)\, d\phi \tag{4.5}$$

The probability $\mathrm{P}_{M|F}$ of misdetection, given that the person walks into the camera's FOV, depends on the image acquisition policy of the camera. Under periodic image acquisition, like the case with the frame rate for initial detection, a person walking through FOV$_i$ is not detected if, for some m:

$$mT_i < t_{in} < t_{out} < (m+1)T_i \tag{4.6}$$

where $t_{in} = t_0 + t_1(\phi, v)$ is the time the person enters FOV$_i$, $t_{out} = t_0 + t_1(\phi, v) + t_2(\phi, v)$ is the time the person exits FOV$_i$, and t_0 is the time the person walks through the door. Since t_0 is, by hypothesis, an outcome of a Poisson process and thus equally likely to be at any time, we can calculate the probability that the condition in (4.6) is verified with the help of figure 4.2 and 4.3.

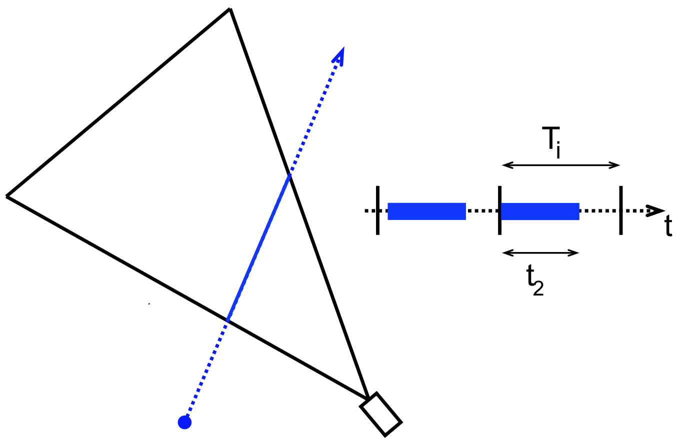


Figure 4.3: calculating probability of *miss* from crossing interval (solid blue)

In the areas of figure 4.2 where $t_2 > T_i$ no miss can happen as the object spends more time in the camera FOV triangle than the frame rate interval is (object c) and by definition, object a is not a miss either. However, objects b can lead to a miss. For such objects, given a speed v, the crossing time at a certain angle is $t_2(\phi, v)$ and is marked as a bold interval on the time line in figure 4.3. The vertical lines represent the detection snapshots taken every T_i seconds. We see that the second object (second bold interval) just arrived exactly after a snapshot was taken and is thus just *missed*. The probability of an object being missed is

$$\frac{T_i - t_2}{T_i} \tag{4.7}$$

Condition (4.6) is therefore satisfied for angles ϕ in Φ with a probability of

$$1 - \frac{\min(t_2(\phi, v), T_i)}{T_i} \tag{4.8}$$

.

Hence by applying the theorem of total probability twice:

$$\mathrm{P}_{M|F} = \int_{\phi\in\Phi}\int_{v} p_v(v)\left(1-\frac{\min(t_2(\phi,v),T_i)}{T_i}\right)\,dv\;p_\phi(\phi)\;d\phi \tag{4.9}$$

Fig. 4.4 shows the relationship between the snapshot period T_i and the miss rate $\mathrm{MR}_{i,d}$ for the situation in Fig. 4.5. This relation should be contained in a look–up table to save power by not computing the various integrals. The Resource Manager can then use this look–up table to decide on a suitable snapshot rate for the camera the look–up table was calculated for.

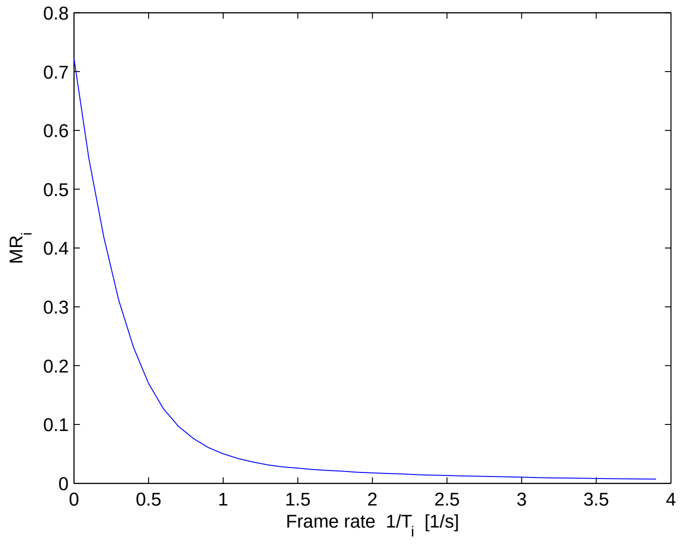


Figure 4.4: Miss rates for initial detection of an object given a frame rate

In practice, the parameters needed to estimate the miss rate are known only with a certain degree of approximation. These parameters include the location and orientation of the camera, as well as the statistical distribution of orientation and velocity. The hypothesis of rectilinear motion at constant speed may not always be accurate. However, uncertainty about the camera geometry can be taken into account by suitably modifying (4.5) and (4.9). Likewise, uncertainty about the actual distributions of v and ϕ can be modeled by increasing the variance of the model distributions.

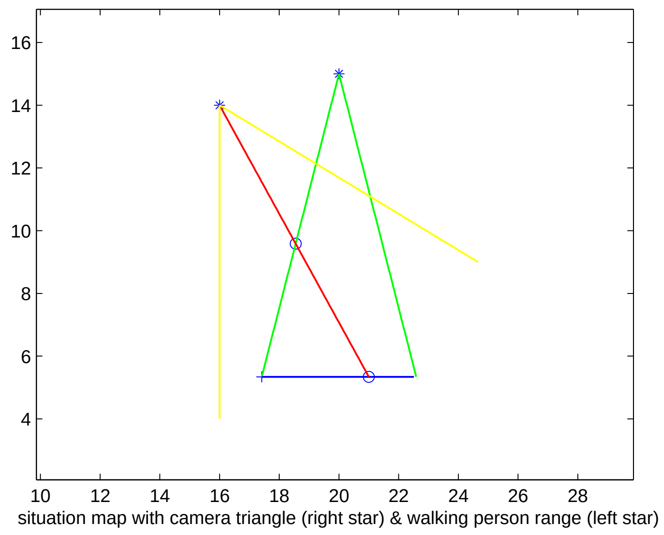


Figure 4.5: Top down view of camera FOV and virtual door for fig. 4.4

4.2 Miss Rate in a Multiple Door Environment

We now consider an environment that is modeled by multiple doors as was shown in figure 2.3. Note that adding cameras to the environment is not incorporated by changing any of the formulas shown so far, instead, each camera will have its own miss rate – frame rate curve solely based on the doors around it. In the last section, we defined the miss rate for door d in (4.2). We now extend this to include multiple (n) doors.

$$\mathrm{MR}_i = \frac{\mathrm{E}[M_{i,d}] + \mathrm{E}[M_{i,2}] + ... + \mathrm{E}[M_{i,n}]}{\sum_d \mathrm{E}[X_d]} = \frac{\sum_d \mathrm{E}[M_{i,d}]}{\sum_d \mathrm{E}[X_d]} \tag{4.10}$$

with the expected number of objects coming out of door d in time span T_s

$$\mathrm{E}[X_d] = \lambda_d \cdot T_s \tag{4.11}$$

$\mathrm{E}[M_{i,d}]$ remains to be calculated according to (4.3), which is repeated here:

$$\mathrm{E}[M_{i,d}] = P_{M|F} P_{F|X} \cdot \mathrm{E}[X_d] \tag{4.12}$$

Keeping the job of the Resource Manager in mind, which will receive the miss rate MR_i based on which it has to find a different frame rate for each camera that balances quality of service, battery life and other factors, we could extend (4.10) to account for the fact, that not all doors might be equally important. One could imagine a hallway with a very secret laboratory and a restroom. Therefore, we add weights w_d for the doors

$$\mathrm{MR}_i^w = \frac{\sum_d w_d \cdot \mathrm{E}[M_{i,d}]}{W_d \sum_d \mathrm{E}[X_d]} \tag{4.13}$$

with

$$W_d = \sum_{all d} w_d \tag{4.14}$$

The miss rate then partly loses its physical interpretation but becomes a measure MR_i^w that gives different doors different weights.

4.3 Learning Environment Parameters after Deployment

Depending on how many snapshots the Resource Manager decides to take after an object was detected and how advanced the object recognition algorithm is implemented, the speed and exact walking path of the object could be recorded. The distributions for $p_\phi(\phi)$ and $p_v(v)$ could then be constantly adjusted based on the recorded data using a receding horizon reaching back a certain time or a number of objects that crossed the FOV.

With good path detection and a model with relatively few doors, the objects could be allocated to a door they probably originated from. Based on this allocation, the λ_d parameters of the doors could be learned. Looking at equation (4.9), we see that the *miss rate* for the single door case is independent of λ_d and learning the parameter wouldn't make sense in that case. In contrast, the *miss rate* (4.10) for the multiple door case is dependent on λ_d, using it to specify the influence of each door on the miss rate. Of course, the weighted *miss rate* (4.13) would also profit from a learned λ_d.

Chapter 5

Wake-up Strategies for Neighboring Cameras

After an object has been detected in one of the snapshots taken at the constant frame rate for initial detection of chapter 4, it makes sense to use this information to wake up neighboring cameras at times, when the detected object is expected to appear in their field of view. This time is calculated from the information recorded by the camera which saw the object last (i-th), general expectation about object behavior, and the known location and orientation of neighboring cameras. In fact, this process of waking up neighboring cameras should continue to take place as woken up cameras detect the object, so that each camera continues to hand off the object to its neighboring cameras.

Taking all available information into account, the Resource Manager needs to decide: (1) which (if any) nearby cameras need to be alerted; (2) how many snapshots each of such cameras should take; (3) what are the optimal times for the snapshots. Intuitively, if a very reliable prediction of the body's motion could be made, only the camera whose field of view will be intersected next by the body's path should be alerted, and just one snapshot should be taken at any time the body is within this field of view. Due to uncertainty in our knowledge of the precise camera and moving body geometry, this prediction will be only approximate, meaning that more than one cameras might have to be alerted, and more than one snapshot might have to be taken.

If the i-th camera, after initially detecting an object, was able to take a couple of snapshots, we can estimate the actual speed, walking direction and also the location where the object was last seen in the i-th camera. The location is used to calculate the distance the object has to move to reach the neighboring cameras and from the distance the time that takes. But even if that information was not available, in a case where the previous camera was just able to take one single shot of the object and might not have been able to exactly localize the object within its FOV, we'd still know that the object was somewhere in the FOV at the time of the snapshot, at a speed that the objects are expected to have due to their nature (people, cars...) with a uniform probability for the direction.

Our strategy is to compute, for each nearby camera of index j, the miss rate MR_j as a function of the number of snapshots N_j it will take, and of the times $\mathbf{t}_j = \{t_{j,1}, \ldots, t_{j,N_j}\}$ at which the snapshots are taken. For each value of N_j, the times $\mathbf{t}_j$ that minimize the corresponding miss rate can be computed, resulting in the optimal (decreasing) sequence of values $\mathrm{MR}_j(N_j)$. Based on this knowledge, the Resource Manager can allocate the number and time of snapshots to be taken by each camera, balancing the need for a low miss rate with the available energy at each node.

Note that besides these snapshots, the woken-up cameras still take the recurring snapshots calculated in chapter 4 to detect so far unseen objects that may have been missed by the nearby nodes. No matter in which of the two types of snapshots an object was seen, the node continues to inform its neighbors.

In the next section, we will calculate the miss rates for single snapshot times in a particular camera, given the position at a certain time and the probability distribution for speed and angle the object is moving. The following section tries to reduce the chance of missing the object in the woken up camera by allocating two snapshots.

5.1 Allocation of a Single Wake-up Shot

We will assume that the location of the body at time t_0 is known exactly, that the body is moving with rectilinear motion at constant speed, and that the distribution of velocity, $p_v(v)$ and of orientation, $p_\phi(\phi)$ of motion are modeled as a truncated gaussian and uniform distribution, respectively, as given in Chapter 2 and used in Chapter 4. Figure 5.1 shows a scenario with a

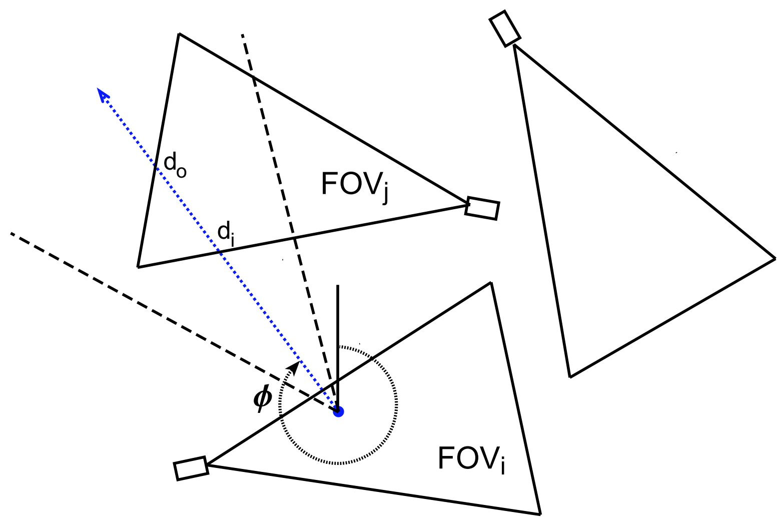


Figure 5.1: Waking up neighboring cameras. An object has been detected at time t_0 by sensor i, which estimated that the body most likely continues its movement within the angular sector marked by the dashed lines.

camera i and j. Note the similarity of the parameters describing the objects position and movement at the point the object was last seen versus position and movement parameters at the *virtual door*. We remember equation (4.4) from chap. 4:

$$\mathrm{MR}_{i,j} = \mathrm{P}_{M|F}\mathrm{P}_{F|X}$$

and define an analogue miss rate for the wake-up times, but with differently defined factors:

$$\mathrm{MR}_j(N_j) = \mathrm{P}_{M|F}\mathrm{P}_{F|X} \tag{5.1}$$

If no snapshot is taken by the j–th camera in response to the event detected by the i–th camera, then

$$\mathrm{MR}_j(0) = \mathrm{P}_{F|X} \tag{5.2}$$

which is the probability that the body will cross FOV_j at some point, and can be computed as by (4.5). To compute $\mathrm{MR}_j(1)$, we first need to express the miss rate as a function of the snapshot time $t_{j,1}$. This requires computing the probability that the times t_{in} and t_{out}, at which the body enters and exits FOV_j, are both before or both after $t_{j,1}$. In symbols:

$$t_{out} < t_{j,1} \text{ or } t_{in} > t_{j,1}$$

and therefore

$$\mathrm{P}_{M|F} = \mathrm{P}(t_{out} < t_{j,1}) + \mathrm{P}(t_{in} > t_{j,1}) \tag{5.3}$$

Using the same symbols as in chap. 4, we observe that:

$$\begin{aligned}\mathrm{P}(t_{out} < t_{j,1}) &= \mathrm{P}(t_0 + t_1 + t_2 < t_{j,1}) \\ &= \int_{\phi\in\Phi}\int_v \mathrm{P}(t_1(\phi,v) + t_2(\phi,v) < t_{j,1} - t_0)\cdot p_v(v)p_\phi(\phi)\;dv\;d\phi\end{aligned} \tag{5.4}$$

Remembering that $t_1(\phi,v) = l_1(\phi)/v$ and $t_2(\phi,v) = l_2(\phi)/v$ we maintain that:

$$\mathrm{P}(t_{out} < t_{j,1}) = \int_{\phi\in\Phi}\int_{\frac{l_1(\phi)+l_2(\phi)}{t_{j,1}-t_0}}^{\infty} p_v(v)p_\phi(\phi)\;dv\;d\phi \tag{5.5}$$

Likewise,

$$\mathrm{P}(t_{in} > t_{j,1}) = \int_{\phi\in\Phi}\int_0^{\frac{l_1(\phi)}{t_{j,1}-t_0}} p_v(v)p_\phi(\phi)\;dv\;d\phi \tag{5.6}$$

Putting it all together, we get

$$\begin{aligned}&\mathrm{MR}_j(1) = P_{F|X}\cdot \\ &\int_{\phi\in\Phi}\left(\int_{\frac{l_1(\phi)+l_2(\phi)}{t_{j,1}-t_0}}^{\infty} p_v(v)p_\phi(\phi)\;dv + \int_0^{\frac{l_1(\phi)}{t_{j,1}-t_0}} p_v(v)p_\phi(\phi)\;dv\right)d\phi\end{aligned} \tag{5.7}$$

As mentioned above, the parameters used to describe an object and its location where it was last seen within FOV_i and from which its path to

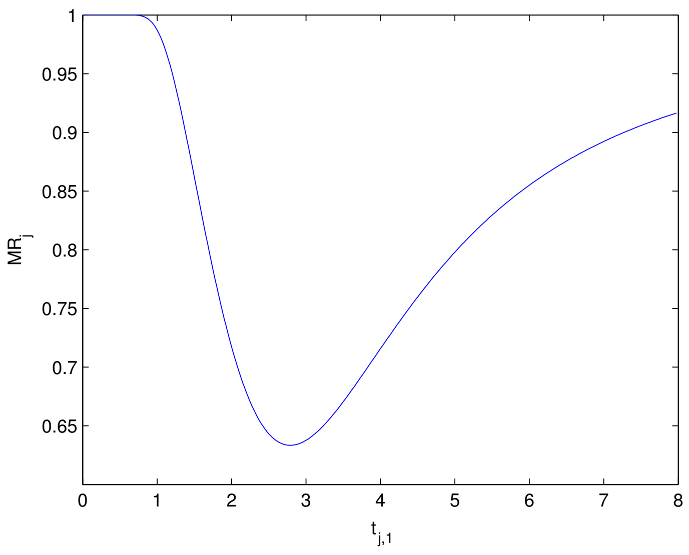


Figure 5.2: The miss rate as a function of the time $t_{j,1}$ at which a single snapshot is taken by the camera. The best wake-up time is around 2.8 seconds.

the next camera was calculated, are the same as used for describing *virtual doors*. Therefore, we can use the same setup of fig. 4.5 as in the last chapter to plot $\mathrm{MR}_j(1)$, which is shown in fig. 5.2 as a function of $t_{j,1}$. In this case the lowest miss rate is $\mathrm{MR}_j(1) = 0.63$ at a wake-up time of 2.8 seconds. Despite minimizing this function leads to the optimal wake-up time, it can be useful to provide not just that single time, but the whole function to the Resource Manager to better handle the following case: It is very likely that several snapshots intended for different objects would be scheduled close to each other as objects often move in groups and also because wake-up calls will arrive from various neighboring cameras. If $\mathrm{MR}_j(1)$ has a flat minimum, the Resource Manager could decide to wake up the camera at a time with slightly higher than optimal miss rate to substitute the newly scheduled and previously scheduled snapshots with a single snapshot. The number of wake-ups would be reduced and therefore energy saved at a just slightly higher miss rate.

5.2 Allocation of Two Wake-up Snapshots

The case of two snapshots ($t_{j,1} < t_{j,2}$) can be dealt with in a similar fashion. In this case, the j–th camera misses the moving body if any one of these disjoint events occur:

$$t_{out} < t_{j,1} \text{ or } t_{j,1} < t_{in} < t_{out} < t_{j,2} \text{ or } t_{in} > t_{j,2}$$

The probability of the first and of the third event above can be easily computed as in the single snapshot case. As for the second event, it is easy to see that:

$$\mathrm{P}(t_{j,1} < t_{in} < t_{out} < t_{j,2}) \tag{5.8}$$

$$= \int_{\phi\in\Phi}\int_{v} \mathrm{P}\left(v < \frac{l_1(\phi)}{t_{j,1}-t_0} \text{ and } v > \frac{l_1(\phi)+l_2(\phi)}{t_{j,2}-t_0}\right)\cdot$$

$$\cdot p_v(v)p_\phi(\phi)dv\ d\phi$$

$$= \int_{\phi\in\Phi}\int_{\frac{l_1(\phi)+l_2(\phi)}{t_{j,2}-t_0}}^{\frac{l_1(\phi)}{t_{j,1}-t_0}} p_v(v)p_\phi(\phi)dv\ d\phi$$

with the understanding that the last integral is null whenever $\frac{l_1(\phi)+l_2(\phi)}{t_{j,2}-t_0} > \frac{l_1(\phi)}{t_{j,1}-t_0}$.

The optimal $t_{j,1}, t_{j,2}$ are shown in Fig. 5.3 for the same camera setup of Fig. 4.5. The best miss rate using two time instants ($\mathrm{MR}_j(2)$) is equal to 0.43, which, as expected, is less than the best $\mathrm{MR}_j(1)$ of 0.63.

For maximal efficiency, the optimal wake-up times should be stored in look-up tables in each camera for a set of likely parameters.

5.3 Constraints Due to Periodic Communication Intervals

We remember from chapter 2 that the cameras can just communicate in synchronized time intervals due to the inability to receive messages in standby mode. This constraint didn't have any influence on taking snapshots at the frame rate of chapter 4 as the snapshot wake-up times are periodic and

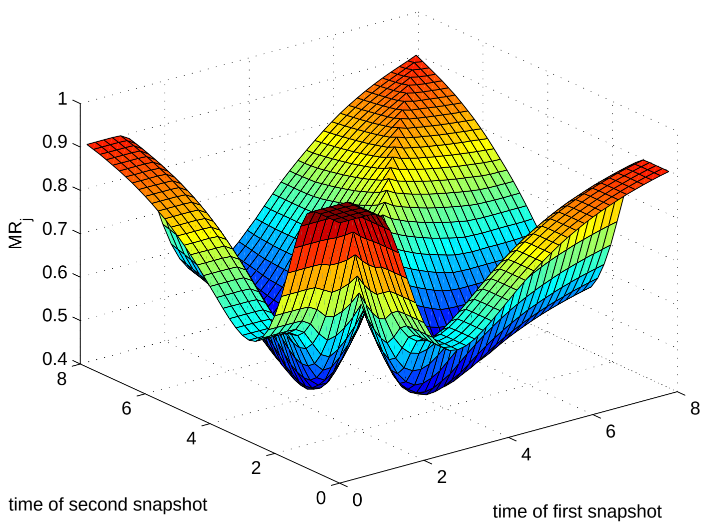


Figure 5.3: The miss rate as a function of the two times at which snapshots are taken by the camera. The graph was mirrored to also include the cases $t_{j,1} > t_{j,2}$

known in advance by all cameras themselves. What will happen is just that the cameras will go to sleep for shorter times than would be necessary for the communication intervals alone. Waking up other nodes instead, at times calculated in this chapter, is just possible if these times are later than the first communication interval after detection. Given an interval T_c between communication slots, the moment of detection is equally likely to be at each moment within T_c as the frame rate for initial detection and communication don't need to be in synch. We therefore have an equal probability for any value between 0 to T_c seconds of duration t after which (4.4) is valid. Before that moment, $\mathrm{P}_{M|F} = 1$ due to the late arrival of the wake-up message. The miss rate is then solely $\mathrm{P}_{F|X}$. Combined with the cases where the wake-up message arrives in time, the miss rate for the one snapshot case now considers network influences and is

$$\mathrm{MRnw}_j(1) = \begin{cases} \mathrm{MR}_j(1) & \text{for } t_1 \in [T_c; \infty] \\ q & \text{for } t_1 \in [0; T_c] \end{cases} \tag{5.9}$$

with

$$q = \int_0^{T_c} \frac{1}{T_c} \begin{cases} \mathrm{MR}_j(1) & \text{for } t_1 \in [t; T_c] \\ \mathrm{P}_{F|X} & \text{for } t_1 \in [0; t] \end{cases} dt \tag{5.10}$$

and with t_1 as the independent parameter of $\mathrm{MRnw}_j(1)$.

Chapter 6

Multi-Door Application and Verification

The last two chapters have shown how to calculate the miss rates, based on which the Resource Manager can choose frame rates and wake-up times for each camera. All miss rate plots so far were calculated for an imaginary environment with one camera and one door, and were not comparable to actual data from real cameras. In this chapter, we will model the entrance of the Baskin Engineering II building at UCSC with multiple doors and place two cameras as shown in fig. 6.1. We then verify the calculated miss rates with data obtained from the actual cameras (fig. 6.2 shows a sample snapshot taken by each camera). The coordinates of the doors and cameras are given in table 6.1.

all objects	$v = 1.28;$	$\sigma_v = 0.3$			
C1	$Xc = 0.41;$	$Yc = 2.10;$	$\alpha_c = 14;$	$\beta_c = 38;$	$w = 12$
C2	$Xc = 0;$	$Yc = 1.54;$	$\alpha_c = 77;$	$\beta_c = 38;$	$w = 2.6$
D1	$Xd = 0;$	$Yd = 10.33;$	$45 < \phi < 170;$	$\lambda = 1/91$	
D3	$Xd = 6.4;$	$Yd = 10.8;$	$200 < \phi < 300;$	$\lambda = 1/60$	
D5	$Xd = 1.28;$	$Yd = 0;$	$340 < \phi < 30;$	$\lambda = 1/166$	
D6	$Xd = 2.43;$	$Yd = 13;$	$160 < \phi < 200;$	$\lambda = 1/203$	

Table 6.1: Parameters used for simulation of Baskin Engineering II entrance (4 door model)

Note that the two elevators visible in the snapshots are represented by a

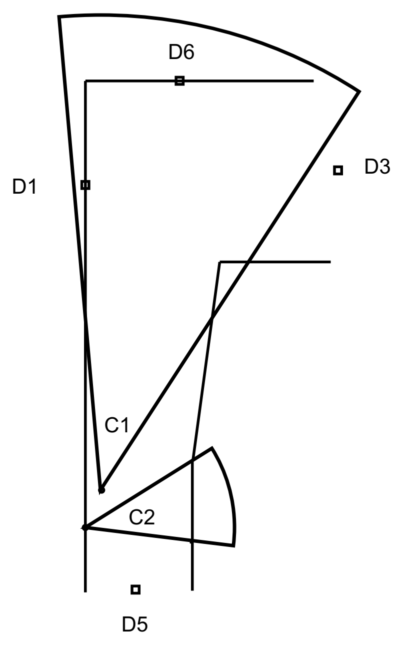


Figure 6.1: Top down view of camera placement and doors at entrance of E2 Building

single virtual door (D6). This was done to demonstrate that the concept of virtual doors introduced in chapter 2 can not only be used to place virtual doors where there are no real doors, but also to model multiple real doors with one virtual door, where applicable.

In section 6.1, the miss rates are calculated for various frame rates and wake-up times and then compared to actual values obtained through subsampling of the snapshots in section 6.2. We then refine the environment model by adding two additional virtual doors and achieve more accurate miss rate estimations. Eventually we combine the constant frame rate for initial detection of objects with wake-up notifications to reduce the miss rate even further.

6.1 Estimated Miss Rates

Figure 6.3 displays the frame rate – miss rate relation for both cameras as calculated with the formulas in chapter 4 using Matlab. The reason that

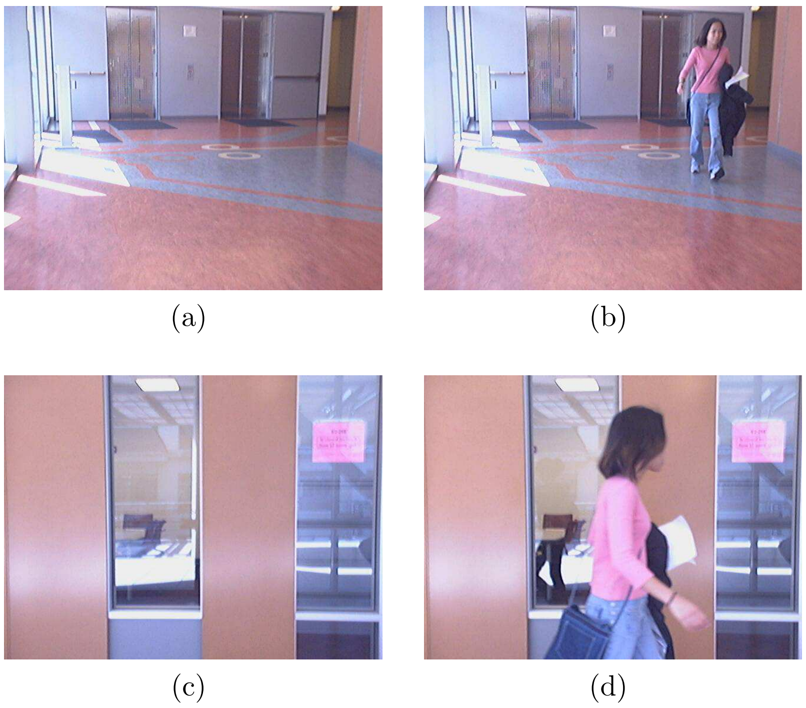
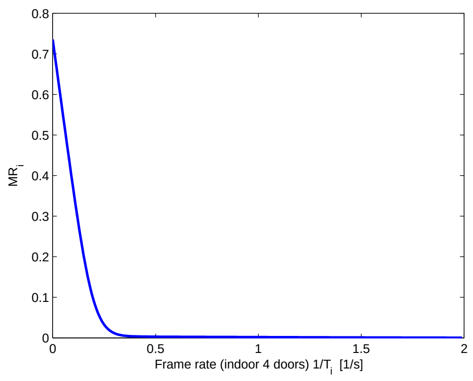
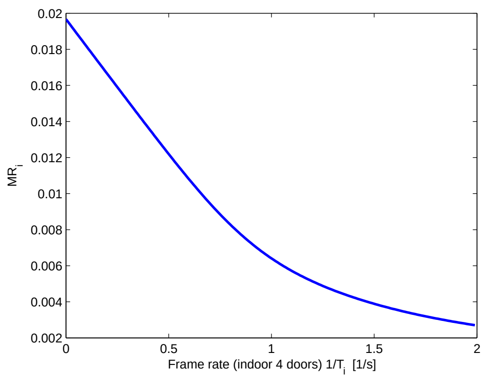


Figure 6.2: Sample snapshots of camera 1 (a & b) and camera 2 (c & d)



(a)

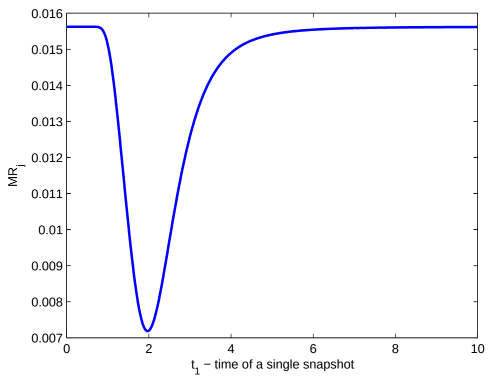


(b)

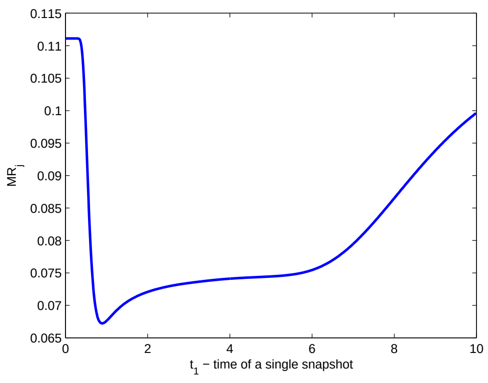
Figure 6.3: Miss rates for given frame rates at camera 1 (a) and camera 2 (b)

the miss rate is not 1 at a frame rate of 0 frames per second is that objects not going through the FOV of that camera are not regarded as miss events according the definition. In other words, the second factor of equation (4.4) is one and the first factor determines the value seen at 0 frames per second. Looking at the curve for camera 1, the Resource Manager would probably choose a frame rate somewhere between 0.2 and 0.3 for a good energy quality trade off.

For being able to calculate the miss rates for varying wake-up times according to chapter 5, we need to specify a point from where the distance to the other field of views are calculated. This point is the location where the object was last seen before it left the field of view of the camera it was detected in (FOV$_i$). To conserve energy, the wake-up times should be stored in a look-up table for various object detection locations. At a later stage in the Meerkat project, there will be software available that, given the camera parameters, automatically detects moving objects in the snapshots and determines their position in world coordinates. For our experiments though, we will not have any information about object location and therefore choose in each FOV, an average location of where objects were last seen (table 6.2). Figure 6.4 shows the miss rates MR$_j$ calculated in Matlab for the two cameras waking each other up. As the FOV of the first camera is so much bigger than the one from the second camera, the range of wake-up times with a similarly good miss rate is intuitively expected to be much



(a)



(b)

Figure 6.4: Miss rates when camera 1 wakes up camera 2 (a) and vice versa (b)

larger for the second camera waking up the first camera than the first waking up the second. The expected behavior is clearly visible in fig. 6.4.

Cam last seen in	X	Y	Angular Range	v	σ_v
C1	2.03m	4.50m	0 - 360°	1.28m/s	0.3
C2	1.28m	2.54m	0 - 360°	1.28m/s	0.3

Table 6.2: Coordinates of last detection

6.2 Measured Miss Rates and Refined Estimations

For comparing the curves calculated with Matlab to curves obtained when analyzing the same scenario in reality, we took snapshots at a frame rate of 5 frames per second with each camera. This frame rate was high enough that no person was able to cross the FOV without being in at least one snapshot. The way the cameras were set up, all objects entering the area under surveillance were captured by at least one camera. As mentioned above, automatic object localization was not yet available in Meerkats, therefore this task had to be done by hand. For this reason, the recording run had to be kept relatively short at a bit more than 30 minutes, producing 9200 snapshot images. Analyzing these snapshots manually, we generated a list of the form shown in table 6.3. The value -1 stands for "not detected in that

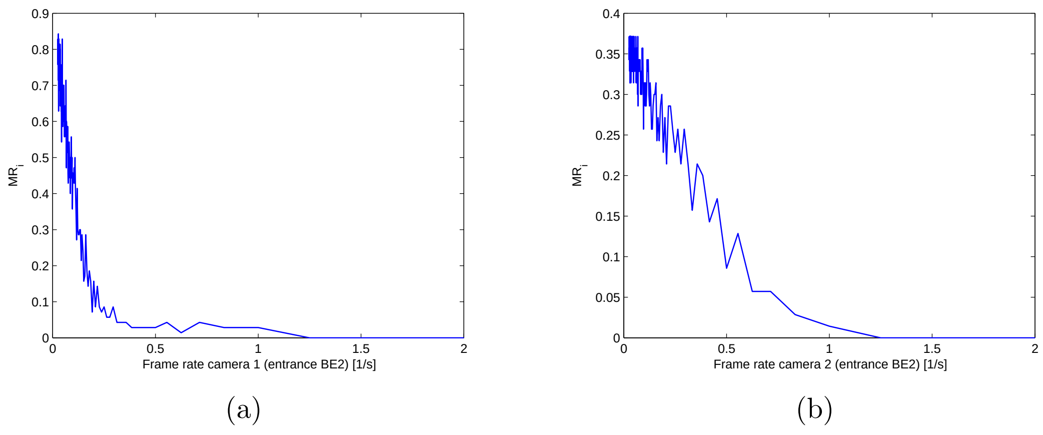


Figure 6.5: Miss rates from simulation for given frame rates at camera 1 (a) and camera 2 (b)

camera". The object location coordinates were not recorded, but the times the object entered and left the FOV.

object	originating door	cam 1 in	cam 1 out	cam 2 in	cam 2 out
1	1	337	367	-1	-1
2	2	342	365	370	375
3	5	368	408	361	364
4	4	691	713	718	724
5	1	723	740	-1	-1
6	4	806	820	826	832
...	...	...	...	...	...

Table 6.3: Beginning of collected data at Baskin Engineering II entrance

6.2.1 Frame Rate for Initial Detection

To simulate taking snapshots at different frame rates for initial detection of objects in the exact same environment, table 6.3 was subsampled with a Java program simulating frame rates of any value between 5 frames per second and 1 frame every 40 seconds for each camera. The resulting real miss rates are displayed in figure 6.5.

While the calculated miss rates for the frame rates were quite accurate for the first camera, the estimation for the second camera is far from acceptable. We recognize that the frame rates, at which the gradient of the miss rate changes the most, correspond well between the computation in Matlab and the Simulation, but the absolute value of the miss rates for the second camera differ a lot. All real objects moving to the lower part of figure 6.2 eventually crossed the second camera due to the hallway walls. In the Matlab calculation though, many of these objects did not cross the second camera and walked straight through where in reality a wall would be. This happened because we assumed the object movement angles distribution covers uniformly all paths between walking from door 1 to 3 or 1 to 6 while the actual distribution would be multi-modal. This flaw in the chosen model for the Baskin Engineering II can be corrected by replacing virtual door 1 with two virtual doors with way smaller angular ranges. We define a new virtual door 1 for the people going to door 3 and a virtual door 2 for people going towards door 6. In an analog way, we replace virtual door 3 by two virtual doors, one for objects going to door 1 and one for objects going to door 6. The improved, 6-door, model now accounts for the fact that objects can't walk a path in between the two directions due to the hallway walls.

all objects	$v = 1.28$; $\sigma_v = 0.3$				
C1	$Xc = 0.41$;	$Yc = 2.10$;	$\alpha_c = 14$;	$\beta_c = 38$;	$w = 12$
C2	$Xc = 0$;	$Yc = 1.54$;	$\alpha_c = 77$;	$\beta_c = 38$;	$w = 2.6$
D1	$Xd = 0$;	$Yd = 10.33$;	$60 < \phi < 110$;	$\lambda = 1/123$	
D2	$Xd = 0$;	$Yd = 10.33$;	$160 < \phi < 175$;	$\lambda = 1/460$	
D3	$Xd = 6.4$;	$Yd = 10.8$;	$255 < \phi < 290$;	$\lambda = 1/80$	
D4	$Xd = 6.4$;	$Yd = 10.8$;	$190 < \phi < 205$;	$\lambda = 1/307$	
D5	$Xd = 1.28$;	$Yd = 0$;	$340 < \phi < 30$;	$\lambda = 1/166$	
D6	$Xd = 2.43$;	$Yd = 13$;	$160 < \phi < 200$;	$\lambda = 1/203$	

Table 6.4: Parameters used for simulation of Baskin Engineering II entrance. Improved model with 6 doors.

Table 6.4 contains the coordinates of the enhanced model with 6 doors and figure 6.6 the new miss rate plots. The miss rate curve for camera 1 is now even more close to the curve from the experiment, but the curve for camera 2 is, despite better than calculated with the 4-door model, far lower than the actually measured values. This difference can partly be explained

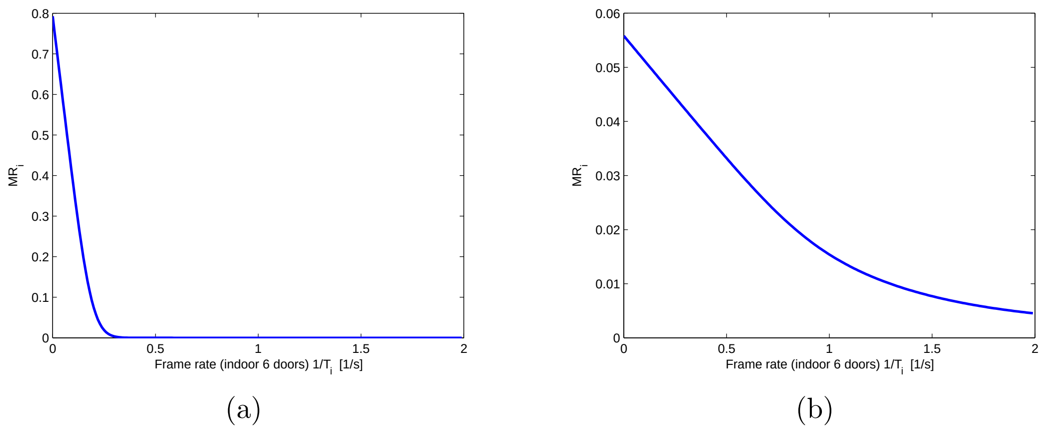


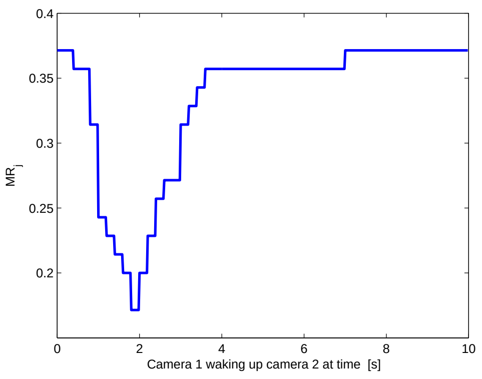
Figure 6.6: Miss rates for given frame rates at camera 1 (a) and camera 2 (b) using the improved environment model with 6 doors

by the difficult case of the small camera 2 located far from most doors, but could also come from a too high actual miss rate, as the latter is based on the relatively small simulation run including only the 70 objects in the 9200 frames captured.

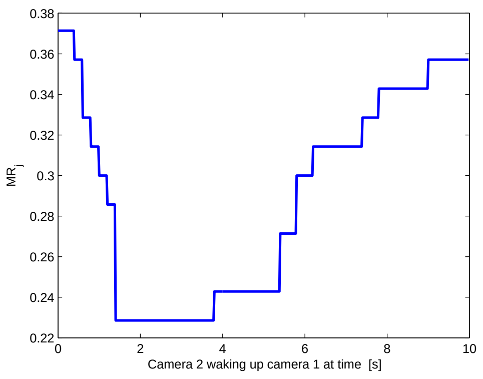
6.2.2 Wake-up of Neighbor Camera

For comparing the calculated miss rates at different *wake-up* times to real world data, the frame rate of the camera to be woken up was set to 0 and the one from the camera which initially detects the object, was set so high, that all objects were detected. For each detected object, a wake-up call was simulated and checked whether exactly the expected object was in the FOV of the woken up camera at the time of the wake-up. Figure 6.7 shows the miss rates MR_j for different wake-up times. The optimal wake-up times according the calculated curves in 6.4 are very close to the real ones from 6.7, but their absolute value differs a lot.

This can be explained by the fact that our model has no knowledge about the direction the detected objects will continue their movement. In table 6.2 we specified an angular range of $0 - 360°$. In contrast, the people walking in the real hallway were limited to a small angular range around 0 and around 180 degrees. We will now reduce the angular range to improve the absolute value of the miss rate. We assume, that once an object is detected, there is



(a)



(b)

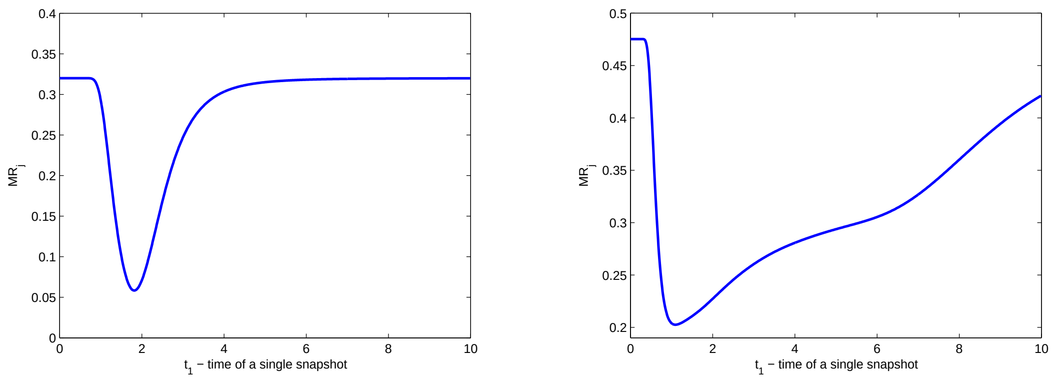
Figure 6.7: Miss rates from simulation when camera 1 wakes up camera 2 (a) and camera 2 wakes up camera 1 (b)

each a 50% chance that the object moves in one of the two directions of the hallway. As there are just 2 cameras set up, just one of the two directions bears the possibility that the object crosses the other camera's FOV. Table 6.5 shows the object detection points with the refined angular ranges and fig. 6.8 the new miss rate plots. The plots were scaled with 0.5 to account for the fact that just 50% of the objects take a path in the specified angular range.

Cam last seen in	X	Y	Angular Range	v	σ_v
C1	2.03m	4.50m	170 - 185° *	1.28m/s	0.3
C2	1.28m	2.54m	300 - 20° *	1.28m/s	0.3

Table 6.5: Coordinates of last detection and reduced angular range of movement. * Objects also choose opposite paths, but they can't lead to a miss as they don't intersect a camera.

Comparing fig. 6.8 to the measured values of fig. 6.7, we see that the calculated miss rates are now much closer to the actual values than was the case in fig. 6.4. The improvement comes from the reduction of the angular range in the object-detection-points to values that account for the hallway walls. One should be aware that the angular ranges used for the calculation are still average values and not yet the values of the actual objects detected. With automatic localization and tracking available in the detecting cameras,



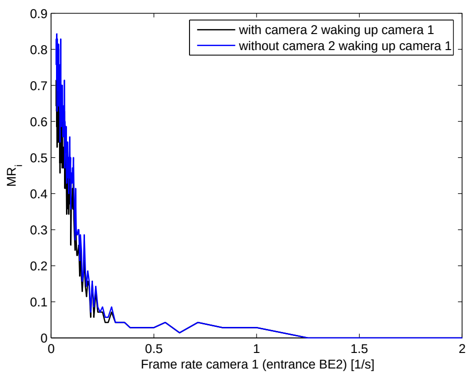
(a) (b)

Figure 6.8: Miss rates when camera 1 wakes up camera 2 (a) and vice versa (b) using the improve environment model with reduced angular range of movement

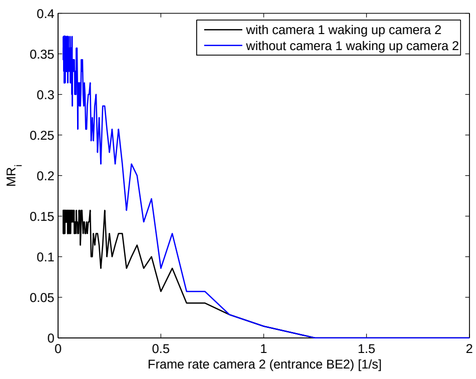
this angular range could be narrowed even more, reducing the difference between the calculated miss rates and the measured values even further.

6.2.3 Combination of Constant Frame Rate with Wake-up Notifications

Figure 6.9 shows how a lower miss rate at given frame rate can be achieved by combining the constant frame rate for initial detection of objects with wake-up notifications. The quality of the wake-ups depends on the frame rate of the camera issuing the wake-up calls, as it can't wake up another camera when it didn't detect the object itself due to a too low frame rate. Therefore, we have set the camera issuing the wake-up calls to a frame rate high enough, so that no objects were missed in that camera. Figure 6.9 shows the miss rate in the woken up camera 1 (a) and 2 (b). The lighter curves show again the same data as was shown in 6.3, which are the miss rates without wake-up calls. The darker curves include wake-ups. We see, that when the woken up camera already takes snapshots at a high frame rate, the wake-ups can't lead to detecting many objects that would have been missed without the wake-ups. The lower the frame rate, the bigger the influence of the wake-up notifications, until at a frame rate of 0 frames per second, the only detections happen due to the wake-ups. The reason for the miss rate of the light curve at a frame rate of 0 not being 1 is again that objects not crossing the FOV of the woken up camera are not considered misses.



(a)



(b)

Figure 6.9: Miss rates from simulation with and without wake-up (black vs blue, respectively). (a) cam 2 wakes up cam 1, (b) vice versa

Chapter 7

Conclusion and Future Work

This thesis approached power reduction in camera sensor networks for surveillance by separating the tasks of initially detecting an object, and informing other cameras about objects in the area. To model the area under surveillance, the concept of *virtual doors* was introduced with which it is possible to model environments as different as open spaces and hallways. The times, at which objects appear through virtual doors were modeled as a poisson point process while speed and direction of movement were modeled by truncated gaussian and uniform distributions. Chapter 4 assigned a miss rate to each frame rate used to initially detect new objects. This miss rate is a *measure*, based on which a Resource Manager can choose a frame rate for each camera while optimizing a cost/utility function containing several criterions such as available energy at each node. Chapter 5 provided an analog measure for wake-up times. The concept was applied to a real environment, the entrance of Baskin Engineering II at UCSC, for which the measures, frame rates and wake-up times have been calculated and compared to values obtained through subsampling data from the actual snapshots.

While the experiment at the Baskin Engineering II entrance provided first results, an experiment with far more than 9200 pictures could be done to compare the calculated values to more averaged measurements, as soon as the automated object extraction and localization is finished in the Meerkats project. The concept could also be applied to objects that tend to change their movement far more, like animals, or far less, like cars on a road as shown in 7.1. On a one way street, one camera could solely rely on the wake-up calls and don't take any initial detection snapshots at a constant frame rate. These photos are from another simulation run done for this

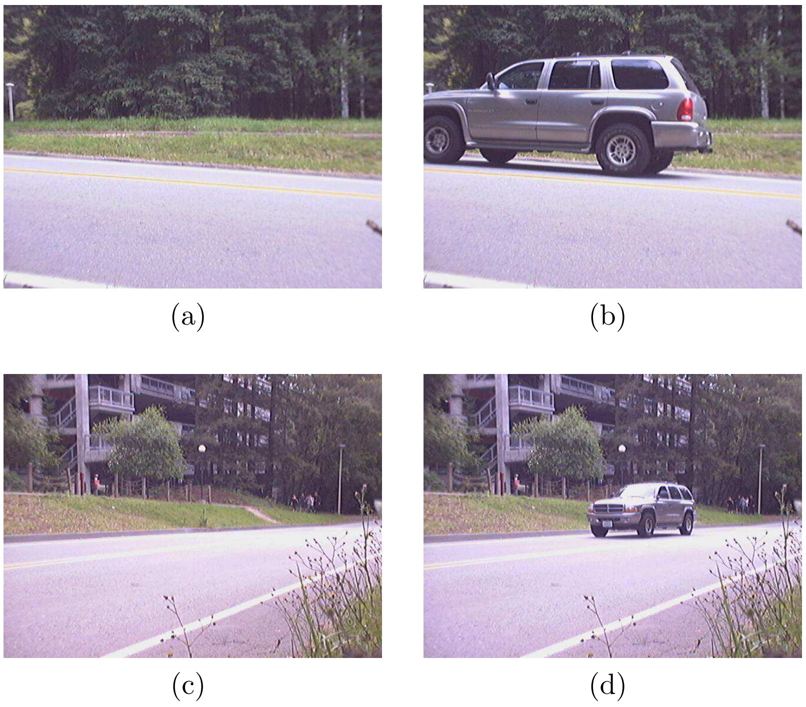


Figure 7.1: (a) & (b): Camera 1. (c) & (d): Camera 2

thesis on the UCSC campus. The car experiment contains more snapshots than the one of the Baskin Engineering Entrance demonstrated in chapter 6, but way less uncertainty and variety in movement angle, which is why the concept was presented with the latter scenario to show the influence of the model accuracy.

In the experiments, all camera FOV triangles were assumed to be equilateral, while, especially when the view is blocked by walls in a non rectangular way, general triangles would be a more close approximation. This would only require changes in the Matlab program calculating the crossing distance of the triangle. Going even further, when using cameras with large angular ranges, even a general triangle becomes inadequate and would have to be replaced by FOV sectors or even circles in the case of omnidirectional cameras.

The results showed, that the concept can be applied to real environments, and that it bears the potential to save a lot of energy by reducing the number of snapshots taken in a significant manner (i.e. 0.3 frames/s compared to 25 frames/s for camera 1 in chap. 6). Combining the frame rate for initial detection with inter-camera wake-ups, the necessary number of snapshots for a given miss rate can be reduced even further.

Even with simple environment models, optimal wake-up times can be calculated accurately and the lowest frame rate values identified, to which the frame rates can be reduced without significant loss of photographed objects. In scenarios with virtual doors placed far from cameras with small fields of view, accurate absolute values of the calculated miss rates depend on detailed modeling of the object movement paths.

Appendix A

Improving Simulation Speed

A.1 Expressing Integral over Normal Curve as a Function of the Tabulated Error Function

$$erf(y) = \frac{2}{\sqrt{\pi}} \int_0^y e^{-t^2} dt \tag{A.1}$$

$$norm_{\mu,\sigma}(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{(x-\mu)^2}{2\sigma^2}} \tag{A.2}$$

substitute $t = \frac{x-\mu}{\sqrt{2}\sigma}$

$$x = t\sqrt{2}\sigma + \mu; \qquad \frac{dt}{dx} = \frac{1}{\sqrt{2}\sigma} \tag{A.3}$$

$$erf(y) = \frac{2}{\sqrt{\pi}} \frac{1}{\sqrt{2}\sigma} \int_\mu^{y\sqrt{2}\sigma+\mu} e^{-\frac{(x-\mu)^2}{2\sigma^2}} dx \tag{A.4}$$

$$= 2\int_\mu^{y\sqrt{2}\sigma+\mu} norm_{\mu,\sigma}(x)dx = 2(\int_{-\infty}^{y\sqrt{2}\sigma+\mu} norm_{\mu,\sigma}(x)dx - \frac{1}{2}) \tag{A.5}$$

$$\int_a^b norm_{\mu,\sigma}(x)dx = \int_{-\infty}^b norm_{\mu,\sigma}(x)dx - \int_{-\infty}^a norm_{\mu,\sigma}(x)dx \tag{A.6}$$

$$= \frac{1}{2}(erf(\frac{b-\mu}{\sqrt{2}\sigma}) + 1) - \frac{1}{2}(erf(\frac{a-\mu}{\sqrt{2}\sigma}) + 1)] \tag{A.7}$$

Exchanging the integral (Matlab quadl function) with the difference of these two error functions reduced the calculation time by approximately a factor of 2.

Appendix B

Useful Representation of Model Geometry

In chapter 4 and 5 we will need to know the time (and therefore the distance) an object coming from a *virtual door* spends in the FOV of a camera as well as the time it takes to reach it. We therefore have to intersect the straight walking lines coming from the *virtual doors* with the camera FOV triangles. In order to do that, we have to choose a parametrization of the FOV as well as of the straight walking path. As the directions on most lines in figure 2.4 do have a physical meaning (walking direction on the straight line, FOV borders away from the camera), we decided for the vector-parameter-form to represent the straight lines and make use of the sign of the parameter. On the straight walking path, the point with the parameter equal to 0 is the virtual door, while positive values will represent the path the object took. Similarly, two of the 3 straight lines used to represent the camera FOV interconnect at the position of the camera at a parameter value of 0. The triangle of the field of view is then expressed as 3 lines of the form

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x_i \\ y_i \end{pmatrix} + \lambda_i \begin{pmatrix} \sin(\gamma_i) \\ \cos(\gamma_i) \end{pmatrix} \tag{B.1}$$

where i are the corner points and the directions γ_i ($0^\circ \triangleq$ north) are chosen that positive parameter values for λ_i make the point $\begin{pmatrix} x \\ y \end{pmatrix}$ follow the side of the camera FOV triangle instead of immediately going away from it.

The following formulas calculate the parameter values λ_1 and λ_2 at the intersection of two such straight lines (straight walking line and one of the

3 lines of the camera FOV triangle). There is of course no such point when the two lines are parallel.

$$\begin{bmatrix} \lambda_1 = \frac{\sin\gamma_2\cdot(y_2-y_1)+\cos\gamma_2\cdot(x_1-x_2)}{\sin\gamma_2\cdot\cos\gamma_2-\sin\gamma_1\cdot\cos\gamma_1} \\ \lambda_2 = \frac{x_1-x_2+\lambda_1 cos\gamma_1}{\sin\gamma_2} \end{bmatrix} \tag{B.2}$$

These formulas B.2 are just valid for $\gamma_2 > 0$, for a $\gamma = 0$ the following formulas have to be used.

$$\begin{bmatrix} \lambda_1 = \frac{\sin\gamma_2\cdot(y_2-y_1)+\cos\gamma_2\cdot(x_1-x_2)}{\sin\gamma_2\cdot\cos\gamma_2-\sin\gamma_1\cdot\cos\gamma_1} \\ \lambda_2 = \frac{y_1-y_2+\lambda_1 cos\gamma_1}{\cos\gamma_2} \end{bmatrix} \tag{B.3}$$

Knowing the maximal possible values for λ_i where the line goes beyond the FOV triangle, it is easy to find out whether the intersection happened at the FOV or on an undefined part of the line.

Bibliography

[BLM$^+$] J. Boice, X. Lu, C. Margi, G. Stanek, G. Zhang, R. Manduchi, and K. Obraczka. Meerkats: A powera aware, self managing wireless camera network for wide area monitoring. in submission to ACM SenSys 2005.

[DB] Anthony R. Dick and Michael J. Brooks. A stochastic approach to tracking objects across multiple cameras. School of Computer Science, University of Adelaide, 2004.

[ES] M. Erdem and S. Sclaroff. Optimal placement of cameras in floorplans to satisfy task requirements. In Proc. OMNIVIS 2004.

[FCK$^+$] Wu Chi Feng, Brian Code, Ed Kaiser, Mike Shea, and Wu Chang Feng. Panoptes: Scalable low power video sensor networking technologies. Department of Computer Science and Engineering, OGI School of Science and Engineering at OHSU, 2005.

[GJV$^+$] L. Gu, D. Jia, P. Vicaire, T. Yan, L. Luo, T. He, A. Tirumala, Q. Cao, J. A. Stankovic, T. Abdelzaher, and B. Krogh. Hierarchical detection and classification in wireless sensor networks. in submission.

[KJRS] S. Khan, O. Javed, Z. Rasheed, and M. Shah. Human tracking in multiple cameras. In Proc. ICCV, 2001.

[MD] A. Mittal and L. S. Davis. Visibility analysis and sensor planning in dynamic environments. In Proc. ECCV, 2004.

[MLN] Raquel A.F. Mini, Antonino A.F. Loureiro, and Badri Nath. Prediction based energy map for wireless sensor networks. Department of Computer Science - Federal University of Minas Gerais, Belo Horizonte, Brazil, PWC 2003.

[MM] A. Mittal and L. Davis. M2tracker. A multi-view approach to segmenting and tracking people in a cluttered scene. International Journal on Computer Vision, 51(3):189 to 203, Feb. 2003.

[PD] Fatih Porikli and Ajay Divakaran. Multi camera calibration, object tracking and query generation. Mitsubishi Electric Research Labs, ICME 2003.

[RDD] Ali Rahimi, Brian Dunagan, and Trevor Darrell. Simultaneous calibration and tracking with a network of non-overlapping sensors. Massachusetts Institute of Technology, Cambridge, MA, 2004.

[RSH] Abhishek Rajgarhia, Fred Stann, and John Heidemann. Privacy sensitive monitoring with a mix of ir sensors and cameras. Proceedings of the Second International Workshop on Sensor and Actor Network Protocols and Applications, Boston, Mass., USA. August 2004, pages 21-29.

[RSPS] Vijay Ragunathan, Curt Schurgers, Sung Park, and Mani B. Srivastava. Energy aware wireless microsensor networks. IEEE Signal Processing Magazine, March 2002.

[ua] unknown author. Pir detector using movement ic and pcb mounted lens. www.hobbyengineering.com - The technology builder's source for kits, components, supplies, tools, books and education.

[WA] Huaming Wu and Alhussein A. Abouzeid. Power aware image transmission in energy constrained wireless networks. Department of Electrical, Computer and Systems Engineering. Rensselaer Polytechnic Institute, Troy, New York, ISCC 2004.

[XL] Yingqi Xu and Wang-Chien Lee. On localized prediction for power efficient object tracking in sensor networks. Department of Computer Science and Engineering, Pennsylvania State University, 2003.

[YHS] Ting Yan, Tian He, and John A. Stankovic. Differentiated surveillance for sensor networks. Department of Computer Science, University of Virginia, ACM SenSys 2003.