Virtual High-resolution for Sensor Networks

Aman Kansal
Microsoft Research
kansal@microsoft.com

William Kaiser, Gregory Pottie, and Mani Srivastava
University of California Los Angeles
{kaiser,pottie,mbs}@ee.ucla.edu

Gaurav Sukhatme
Department of Computer Science
University of Southern California
gaurav@usc.edu

Abstract

The resolution at which a sensor network collects data is a crucial parameter of performance since it governs the range of applications that are feasible to be developed using that network. A higher resolution, in most situations, enables more applications and improves the reliability of existing ones. In this paper we discuss a system architecture that uses controlled motion to provide virtual high-resolution in a network of cameras. Several orders of magnitude advantage in resolution may be achieved, depending on tolerable trade-offs. We discuss several system design choices in the context of our prototype camera network implementation that realizes the proposed architecture. We also mention how some of our techniques may apply to sensors other than cameras. Real world data is collected using our prototype system and used for the evaluation of our proposed methods.

Categories and Subject Descriptors

C.4 [**Computer Systems Organization**]: Performance of Systems; C.2.4 [**Computer Systems Organization**]: Computer Communication Networks—*Distributed Systems*

General Terms

Algorithms, Performance, Reliability, Measurement

Keywords

mobile systems, network protocols, coverage, actuation, motion coordination, spatial resolution, mobility control

1 Introduction

The spatial resolution at which a sensor network provides coverage is an important consideration in its design since this resolution may determine if a desired application can be realized or not using that network. In particular, for an image sensor network, the image resolution determines how finely the covered space is being photographed. Suppose the density at which the cameras are deployed and the image resolution of each camera is such that a unit length in real space, placed at any point in the area covered by the network, maps to P pixels in the image. Then, we define the coverage resolution[1] as $P pixels/m$. Different points in the covered area may be at different distances from the respective nearest camera and coverage resolution may be defined to be that provided at the worst case point covered by the network. Figure 1(a) shows an example scene desired to be covered by a camera network. Suppose the image data is to be used by a sensor network application interested in photographing the vehicles in the scene. A small portion of the scene where a vehicle is present is shown at $15 pixels/m$ resolution in Fig. 1(b). This resolution may be sufficient for applications interested in detecting vehicles, such as using motion detection, in the scene but not for applications interested in recognizing the vehicles. Figure 1(c) shows[2] a small section of the same field of view with the coverage resolution increased to $200 pixels/m$. Such a resolution may enable an application that is interested in recognizing the vehicles present in the scene. The simple example clearly shows that increasing the resolution may enable newer applications. Given a finite set of sensing resources, it is thus important to provide the highest coverage resolution feasible with those resources.

(a) Scene (b) 15 pix/m (c) 200 pix/m

Figure 1. Resolution determines feasible data processing.

There are many ways in which the resolution may be improved. Given the feasible maximum sensor density, signal processing may be used to combine images from multiple cameras to obtain a higher resolution image, sometimes referred to as a super-resolution image [1]. Another method to

[1]Pixels may be rectangular rather than square making a unit length in space map to different number of pixels depending on whether the length is considered aligned horizontally or vertically: we use horizontal alignment for defining coverage resolution.

[2]The image resolution in the printed paper may vary from the resolution of image data used in the figure.

increase resolution with a given set of sensing resources is to use motion, and we discuss this latter approach in this paper. We refer to the high resolution provided using controlled motion as *virtual* since there are certain trade-offs involved compared to high resolution obtained using higher resolution sensors.

Motion helps improve sensing in at least two ways:

Adaptive resource allocation: Motion may be used to adaptively focus sensing resources on regions of interest. For instance, suppose the camera used to capture the image shown in Figure 1(a) has pan, tilt, and zoom motion capabilities. Then the total sensing resources used to obtain the image shown in Fig. 1(a) may be allocated, using those limited motion capabilities, to the small region shown in Fig. 1(c). The application using the image data is interested only in regions where phenomena of interest, such as the vehicles in this example, are present and to that application, motion makes it appear that the sensor network is providing a $200 pixel/m$ equivalent spatial resolution even though significantly fewer sensing resources are being used.

Avoiding obstacles and overlap: Practical deployment environments rarely ever provide an isotropic sensing medium without obstacles. Clearly, a sensor can use motion to reduce the occlusions to its view due to obstacles in the medium. Such use of motion assumes that the sensor can obtain information about the obstacles, either from its own data or through external means. Further, when multiple sensors are present they may use motion to minimize overlap areas among their coverage, and hence potentially focus on smaller regions, providing a high resolution coverage within those smaller regions.

We focus our discussion on a specific type of motion capability in the cameras, namely the ability to pan, tilt, and zoom. Such cameras are sometimes referred to as active cameras in computer vision or motile sensors in robotics. There are several reasons which motivate us to restrict to this type of motion:

1. The navigational overheads of such constrained motion and very low compared to navigating an unconstrained mobile robot, which requires significant hardware and software resource overheads.
2. The energy requirements for constrained motion are lower than unconstrained mobility since the bulkier components such as the battery, motors and processing platform can stay stationary while only the transducer has to move.
3. Such motion is feasible in tethered nodes such as high bandwidth video sensors where wireless communication may limit data quality.
4. If the tolerable delay in sensing is small, then only a limited range of motion may be feasible within the actuation time allowed and small motion capabilities are sufficient to exploit the delay available.
5. In certain applications the intrusion into user space due to unconstrained mobile nodes may not be acceptable while small motion such as pan, tilt, and zoom can be incorporated with negligible intrusion.

A detailed discussion of the sensing advantage due to various types of motion capabilities, such as motion along a one dimensional track or cable, or unconstrained robotic motion, along with pan, tilt, and zoom methods for increasing sensing performance appears in [2].

1.1 Key Contributions

The goal of this paper is to explore the issues in system design when motion is used for providing virtual high resolution in a sensor network of cameras. A review of the prior work reveals that the approach proposed in this paper has not been considered before for providing high-resolution coverage.

First, we discuss if and when it helps to use motion as opposed to alternative methods, such as, using a higher density of static sensors to improve the resolution at which the area of interest is covered. We discuss the trade-offs involved, such as the time it takes for the motion actuators to provide the virtual high-resolution coverage. We systematically characterize this time as the actuation delay, which is a crucial design parameter, and we discuss it in depth.

Second, in scenarios where the use of motion capabilities is a preferred strategy, we discuss methods to efficiently use motion. We develop a system architecture that can effectively exploit motion capabilities and adapt to the environment. For instance, since actuation delay is a key tradeoff in system design, our system architecture includes methods to use motion in a manner such that sensing delay is minimized. The sensing performance and the design of motion methods can benefit from additional information about the deployment environment, such as the characteristics of the sensing medium and the spatial distribution of the phenomenon of interest. Our system architecture enables the network to exploit such information in a distributed manner. While our design is focused mainly on camera networks, we briefly mention the use of motion to create virtual high resolution for other sensors that measure the phenomenon at just the point at which they are present, such as a temperature sensor.

Finally, we realize our proposed system architecture in a prototype system. We also evaluate our methods on a realistic application scenario, using event data collected from a deployment at our campus. The camera data stream used for the test is also publicly shared via our CVS repository [3].

1.2 Prior Work

The use of motion to enhance coverage in sensor networks has been considered before, such as in [4]; however, random, as opposed to controlled controlled, motion was considered. Controlled motion has also been studied for the optimal deployment and coverage maintenance for sensors in distributed robotics [5, 6, 7, 8, 9, 10], computer vision [11, 12, 13], and sensor networks [14, 15, 16]. The methods we propose are complimentary to those and may be used after an optimal deployment has been found using those methods. Unlike deployment methods which typically need not be concerned with motion delay, our methods address the actuation delay trade-off in providing high-resolution sensing.

The problem of improving sensing performance in networks of cameras with limited motion capabilities has been considered before in [17, 18, 19, 20, 21] among several other works. However, these works are distinct from ours. They use motion to allocate the available sensing resource successively to different locations. The motion objectives are distinct from our objective of virtual high-resolution, and the motion is typically used reactively to track certain events detected by the camera. Also, the actuation delay considerations are very different for our approach. The differences will become more evident through the description of our proposed methods.

2 Design Considerations

In this section we provide an overview of the issues involved in designing a system that uses motion to improve the sensing resolution.

2.1 Design Issue: Whether to Use Motion

The first design issue of interest has to do with the use of motion itself. There are several trade-offs involved in choosing between a static network and a network with motion capabilities. Some of these are:

2.1.1 Feasible Resolution

The deployment environment may limit the density of static sensors that may be deployed. This limit may arise from various reasons. The ability to provide mounting supports may be restricted to a limited number of locations within the environment being monitored. The size of the sensor node limits the density of deployment. A very high density of sensors may interfere with the phenomenon itself. Additionally network capacity, such as the channel bandwidth in a wireless system, may limit the density of nodes. The provision for energy may impose another limitation. If a wired energy infrastructure is provided, the density at which the cabling can be installed will limit the node density. For systems using environmentally harvested energy, the amount of energy available in the environment may limit the number of nodes.

Consider for instance the resolution feasible using a finite set of sensing resources with and without motion. The image shown in Fig. 1(a) is obtained using a 1 Megapixel (1MP) camera. The resolution of coverage over the region of interest is: $40 pixels/m$ at the near end of the road and $7 pixels/m$ at the far end of the road in the scene. In the image shown in Figure 1(c), the same 1MP sensing resource is allocated to a small region achieving $200 pixels/m$ resolution over that region. Providing such high resolution coverage over the entire scene using only static cameras, mounted at the same location, requires sensing resources of 784MP[3].

Let us compare what motion may achieve. Suppose the 1MP camera used to obtain the image is equipped with pan, tilt, and zoom motion capabilities. Then, a zoom of 13X along with a small pan and tilt motion is sufficient for this 1MP camera to capture the image shown in Fig. 1(c) by focusing its sensing resources selectively on a small region. These motion capabilities may be used to get high resolution at any subregion of the scene shown in Fig. 1(a) and thus motion has provided a virtual 784MP resolution using only 1MP sensing resource. There are several issues in using the latter alternative such as the delay in carrying out the pan, tilt, and zoom steps and the issue of finding out the regions of interest. These issues will be discussed at length in the subsequent sections. For situations when those issues can be satisfactorily addressed, using motion is a significantly better alternative.

[3]Deploying cameras closer to the road could enable the use of much lower resolution cameras such as used in red-light violation monitoring systems, but we assume that we are not allowed to install our equipment on the city road infrastructure.

Thus, for the application scenario of interest, if the required resolution cannot be achieved with a static system (such as if 784 1MP cameras are impractical to be deployed in the above example), but can be achieved with a mobile network, clearly, the use of motion is to be preferred.

There may be other applications where the resolution achieved by a static network of low power cameras such as developed in [22] may suffice. In some instances, the area to be covered may be so small, such as a known choke point in the field of view, that high resolution can be provided using a static cameras. Again, motion may not be required to increase resolution in that scenario.

2.1.2 System Deployment and Operation Cost

A second concern in choosing to use motion may be the system cost. Suppose the number of nodes required in a static network is N_s and the number of mobile nodes to get comparable coverage is N_m. Excluding certain applications such as those where the sensor nodes are scattered from an airplane over an open field, most applications, particularly those in indoor and urban environments, require each sensor node to be individually deployed at a specific location, often manually. For instance, deploying a camera requires mounting it on a wall or ceiling, installing a dome or cover, and connecting it to a wired or wireless communication network. For the current generation of cameras, deployment also involves providing a power connection, although batteries or environmentally harvested energy may power future generations of low power cameras. Suppose the deployment cost per node is c_{dep}. Suppose the cost of data processing at the back-end for data received from a node is c_p. These two costs are likely to be same for both static and mobile nodes. Suppose also that the cost of a static node including maintenance for the duration of deployment is c_s. Denote the corresponding number for a mobile node as c_m. This number depends on the type of motion capabilities included in the mobile node. The total system cost for the static case becomes $N_s(c_s + c_{dep} + c_p)$ and that for the mobile case becomes $N_m(c_m + c_{dep} + c_p)$. Noting that the node costs c_s and c_m are likely to diminish with advances in camera sensors and MEMS based motion technology, in addition to the fact that c_{dep} and c_p are same for both types of nodes, leads to the conclusion that the trade-off will be determined primarily by values of N_s and N_m. For the current generation cameras, since the difference between c_s and c_m is within an order of magnitude while the difference between N_s and N_m is greater than an order of magnitude, typically several orders of magnitude for examples considered in this paper, again the trade-off is dominated by N_s and N_m.

2.1.3 Actuation Delay

Another difference between a static and mobile network is that while in a static network, the sensing is instantaneous, it is not so in a mobile one. Since the mobile camera takes a finite non-zero time for the actuating to the region of interest, there is a delay before the application receives the desired high-resolution data. We call this delay the actuation delay. The advantage in resolution depends on the tolerable actuation delay; the motion advantage disappears if the tolerable delay is zero. Hence, this is a crucial issue in deciding whether motion may be used.

The actuation delay scales better than linearly with the number of cameras saved. Recall the earlier example where 784 static 1MP cameras were required to obtain the desired resolution. A naïve approach can achieve a linear scaling in delay. For instance, a single camera may be actuated to perform a raster scan visiting the appropriate pan, tilt, and zoom poses to successively cover the small regions covered by the 784 cameras. However, a better strategy such as using the single 1MP camera to cover the entire scene at low resolution, applying a low resolution algorithm to detect regions of interest in the field of view and actuating only to cover those regions, reduces the delay significantly. In this case, the camera does not visit all 784 poses but only a few of those, depending on the number of detected events.

2.2 Design Issue: How to Use Motion Effectively

The second design issue is to provide a system architecture that exploits motion capabilities effectively to provide the best possible sensing performance. As mentioned before, the method used to increase the resolution using motion is to detect the phenomenon of interest using low resolution coverage and then selectively concentrate more sensing resources on regions where the phenomenon is detected, providing an equivalent high-resolution coverage to the overlying application using the sensor network.

Obviously, one of the system components required in realizing this approach is a motion control method to carry out the motion required each time a phenomenon detection occurs. A simple example of this motion occurs in a camera with pan, tilt and zoom capabilities: the camera may zoom in to the pixel location in its field of view where a phenomenon of interest is detected. More sophisticated motion strategies may be used for other objectives such as providing coverage by multiple cameras when a phenomenon of interest is detected by one of them. The exact requirements are application specific, and some methods have been explored for such motion control [18, 19, 20, 21]. Further methods may be developed to the scenario of multiple events being detected in the field of view of a camera, such as by prioritizing between events and scheduling them sequentially.

Somewhat less obvious perhaps, and not previously studied to the best of our knowledge, is the need for a system architecture that allows the system to use the motion capabilities effectively for the above mentioned motion control component. We focus our attention on this latter aspect.

Before the movement of nodes to provide high resolution coverage takes place, the phenomenon must be detected and the system is thus required to provide a good detection performance. Further, when the movement occurs, the actuation delay is a key consideration, particularly when the mobile network approach is compared to the alternative of using a higher density of static sensors to achieve the same resolution. Clearly, this delay should be characterized. The system architecture should use motion capabilities in such a way that the actuation delay is small for phenomena occurring at any point in the area covered by the network. The limits on tolerable delay will also limit the range of motion that may be used. The system architecture should provide for minimizing the actuation delay.

The system architecture should also attempt to exploit any information available about the deployment environment. Specifically, the following two types of information may be useful in controlling motion: a map of obstacles in the sensing medium and the expected spatial distribution of the phenomenon being sensed. It is desirable that the system architecture include components in the system to learn such information and that it allows the network to gather and use such information in a distributed manner.

2.3 Assumption

In designing our system architecture in view of the above design issues, we have made the following general assumption. We assume that each sensor knows its location. This assumption is required only for improving the sensing performance through coordination among multiple nodes. A local rudimentary strategy to use motion without any coordination among sensors is also mentioned, and while it does not require the use of location information, such a strategy performs significantly worse than one with coordination. The assumption about the availability of location information is reasonable in many applications since the location information is required to geostamp the sensor data itself. Several localization strategies have been proposed for sensor networks [23, 24] in general and for networks of cameras in particular [25, 26]. For the type of motion being considered in our work, even a deployment time record of node installation locations suffices.

Other specific assumptions in making certain decision choices are discussed with the system design in the next section.

3 System Design

We discuss our system design and show how it addresses the issues mentioned earlier, in addition to describing other design choices made in our system.

3.1 Example Application

While most of our design choices apply generically to any camera network where resolution is to be increased, some of the analysis is performed in the context of the example application mentioned in section 1, and specified concretely here. Consider the access road to a building shown in Fig. 1(a). This image is captured from a camera mounted on a 2nd floor patio of the building itself. Suppose the entire sealed road area in this scene is to be photographed at a resolution good enough to visually recognize a vehicle, say $200 pixels/m$. The vehicle license plate, if captured in the image, will be legible at this resolution. However, cameras may be installed only within the building premises and not on the roads themselves since the building owners may not have jurisdiction over the road infrastructure or the sidewalks.

We are not concerned with the actual application level image processing algorithms but only with providing the high resolution image for such processing.

3.2 Using Motion

Let us first look at the issues described in section 2.1, concerned with whether motion should be used. For the application scenario considered, the calculations from section 2.1.1 showed that using motion could yield close to three orders of magnitude advantage in the number of cameras required. While this is promising, we must also evaluate the delay penalty involved.

Rather than having the camera perform a raster scan among all required poses, we use the alternative approach of first detecting where the regions of interest are so that actuation is required only for providing high resolution at those locations. There are several applications where detection of an event of interest may be carried out at low resolution. For instance faces can be detected using color detection [27], which requires a lower resolution than required for recognizing. Humans or vehicles in a scene may be detected using motion detection, whereas high resolution coverage may be required for the actual surveillance application logging a high resolution image of the visitors. In other applications, an alternate sensor modality may be used for detection. For instance, magnetic sensors may be used to detect vehicles and then image sensing resources may be allocated to the required region. An acoustic beam forming technique may be used to detect where an animal call originates [28] for an ecology application, or where a shooter is present for an urban security application [29], and then high resolution imaging may be directed toward that location.

Suppose the first approach, of using the image data itself at low resolution, is applied to determine the regions of interest. It is necessary that a detection algorithm that works at a resolution lower than the resolution required for final sensing application is available. Suppose the detection algorithm works at resolution R_{det} pixels/m and the final sensing application requires resolution R_{sense} pixels/m. The available zoom range then is the ratio $z = R_{sense}/R_{det}$. Suppose the field of view of the camera at its widest zoom setting is θ_{fov} in the horizontal direction. Given the pixel resolution of the camera sensor, the range R up to which the resolution R_{det} is provided may be computed. Then, the volume, V_d, over which the camera is able to provide detection resolution may be modeled as the pyramid shown in Fig. 2. Suppose, the camera hardware provides a zoom range z_{hw}, greater than z. Suppose it provides a pan range θ_{pan} such that $\theta_{pan} \geq \theta_{fov}$. Similarly assume that the hardware tilt range is greater than the field of view angle in the vertical direction. Then, if an event of interest is detected at any point within volume V_d, high resolution may be provided at that location. The smaller pyramid V_s in the figure depicts the smaller volume covered at R_{sense}, at some instance. The actuation delay for this scenario is the worst case delay for the camera to actuate from its wide angle setting covering V_d to some pose so as to cover a volume V_s when some interesting event is detected. The actuation delay depends on the hardware used to implement the motion capabilities of the node. Suppose the tolerable actuation delay is t_a. The tolerable delay may not be sufficient to visit points in the entire V_d and let us denote the volume that can actually be visited within t_a as $V_d(t_a)$. Then, the number of cameras required is reduced by a factor $G = V_d(t_a)/V_s$. We evaluate the trade-off between delay and sensing resource advantage for the hardware used in our prototype system.

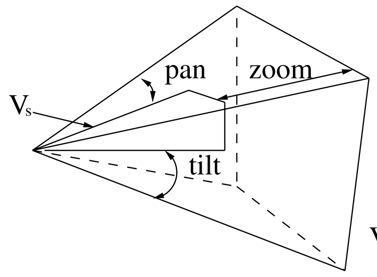


Figure 2. Model for detection volume and sensing volume.

In our prototype, we use motion detection to detect a phenomenon. Given that the application is interested in photographing vehicles, assuming that the minimum vehicle width is $1.5m$, and also assuming that motion detection can work reliably if the one of the moving object dimensions is at least 15 pixels, we see that $R_{det} = 10 pixels/m$. For the required $R_{sense} = 200 pixels/m$, this yields $z = 20X$. The pan, tilt, and zoom (PTZ) camera we use in our system [30] has a zoom range greater than this. The available actuation capabilities of the camera are compared to those required in Table 1. The pan and tilt speeds for this camera are documented in the data-sheet. The zoom time is not documented and is not linear in the zoom factor. We experimentally measured the zoom time (Fig. 3). Using the documented pan and tilt speeds along with the measured zoom time, we can calculate the volume $V_d(t_a)$ that may be accessed within delay t_a. For the fixed volume V_s, calculated based on R_{sense}, the gain in sensing resources, G is plotted in Fig. 4 for varying t_a. The value of G reaches as high as 8000 when allowing an actuation delay of $t_a = 2.5s$ which exploits the full V_d feasible at $z = 20X$. The graph shows that even for moderately small values of t_a, the value of G, plotted on a log scale, is significant, crossing an order of magnitude at about 0.7s. Note that only a small portion of the pan and tilt capabilities are being exploited here; further advantage could have been possible if the entire ranges were used.

Capability	Required	Available
Zoom	z=20X	z_{hw} = 25X
Pan	$\theta_{fov} = 45°$	$\theta_{pan} = 370°$
Tilt	30°	115°

Table 1. Required actuation ranges and hardware capabilities.

The delay can be further reduced by using more than one PTZ camera. For instance, if 4 cameras are installed instead of one, each camera is responsible for only a quarter of the

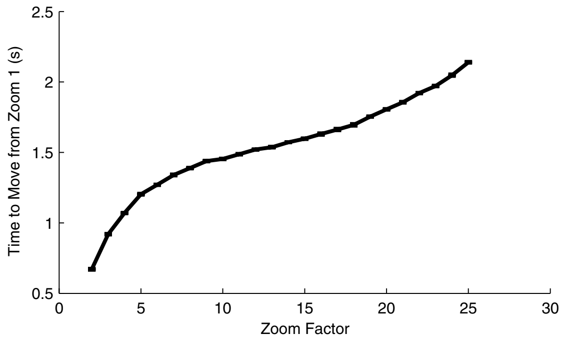


Figure 3. The time to actuate from the widest angle to various zoom factors for the camera used in our prototype.

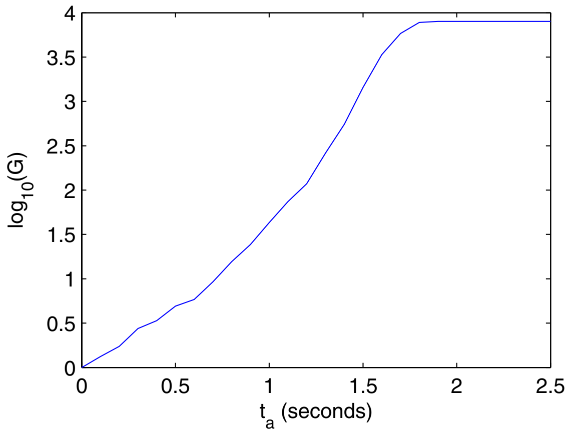


Figure 4. Coverage advantage with varying actuation delay.

V_d and a lower actuation time is sufficient to cover it. In our prototype system, we use four cameras based on equipment availability. In actual production, the number of cameras may be determined based on the desired delay tolerance and the expected number of simultaneous events in the scene.

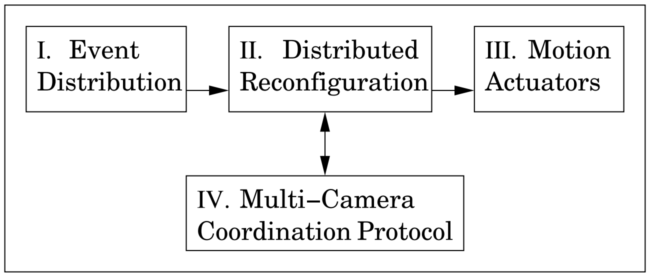
The multiple orders of magnitude savings in sensing resources with a modest delay penalty motivates the use of controlled motion in our prototype. Providing the networking, power, and mounting infrastructure for 4 cameras is far more reasonable than for 784 cameras.

3.3 System Architecture

We present the design of our system architecture and the various choices and trade-offs involved. Fig. 5 shows a block diagram of the various software components.

Block I, labeled event distribution, denotes any methods available in the system to learn the spatial distribution of interesting events.

Block II, labeled distributed reconfiguration, represents the algorithm used to choose the appropriate pan, tilt, and zoom setting at each camera such that actuation delay, when some event is detected, is minimized. We refer to these reconfiguration methods are proactive motion control methods to keep them distinct from the motion control methods used



I. Event Distribution
II. Distributed Reconfiguration
III. Motion Actuators
IV. Multi–Camera Coordination Protocol

Figure 5. System software architecture.

reactively after an event is detected, for zooming in to that particular event and sensing it at high resolution. We describe the proactive methods developed for our system in section 3.3.1

Block III, labeled motion actuators represents the motion capabilities available in the system. These motion capabilities are used both by the proactive method to choosing a good configuration for the network and by the reactive methods for providing the high resolution sensing when an interesting event is detected. For instance, we saw in Table 1 that the hardware used in our prototype has higher actuation ranges than required for providing high resolution at locations where events may be detected. For instance, while 20X zoom is required, the available zoom is 25X and hence the remaining 5X zoom is available for the proactive methods. Similarly, a large portion of the pan and tilt capabilities are available for the proactive phase. The camera used allows controlling its motion capabilities via sending it command packets in HTTP format with hardware specific values for various motion settings. We developed a driver, in Java, with a programmer friendly interface that accepts the required pan, tilt, and zoom motion steps in decimal values. Our driver takes care of converting these to hardware specific format and packetizing them as per the protocol required by the hardware.

Block IV, denoted multi-camera coordination protocol, represents the network protocol developed for communication among multiple cameras for supporting the distributed reconfiguration algorithm of block III. This protocol is described in section 3.3.2.

A key contribution of this software architecture is that it explicitly provides a method to address the delay trade-off that results due to the use of controlled motion. This architecture is distinct from previously designed active camera networks that used motion to patrol larger areas [17, 19, 21, 18] that required constant motion for the patrol. On the other hand, the low resolution detection phase that we use does not require constant motion. Further those methods are mostly designed for the reactive phase of tracking events after being detected. That is distinct from the high-resolution objective we present. Also, the previous works do not proactively configure the system to reduce the actuation delay.

3.3.1 Coordination Algorithm

The objective of the proactive motion coordination algorithm is to set the pan, tilt, and zoom pose at each camera, i.e., the overall network configuration, such that the actuation delay required to zoom in to any point in the area covered by the network is minimized. At the same time, the detection capability should be maximized, which involves having coverage over a large area. To this objective, we first define a metric which quantifies the utility of a given network configuration.

Let the region to be covered by the network be denoted by A and let p be a point in A. For an event detection at point p, using the motion capabilities chosen for our system, the actuation delay depends on the time taken by the camera to pan, tilt and zoom for capturing a high-resolution image of point p. Denote this time by $\tau(p)$. Since the pan and zoom are driven by separate motors and can occur in parallel, the time required is:

$$\tau(p) = max\{\delta\theta_p * \omega_p, t_p(\delta_z)\} \tag{1}$$

where $\delta\theta_p$ is the pan angle to be moved to direct the sensor at p, ω_p is the angular pan speed, δ_z is the change in zoom required for achieving the required classification resolution at p, and $t_p(\delta_z)$ is the time required for changing the zoom Fig.3. The tilt time can be considered similarly as the pan time and we skip mentioning it for brevity.

For characterizing the detection performance, several possible metrics exist, beginning with a purely geometric coverage metric which quantifies the area in line of sight of the sensors, to more sophisticated estimation theoretic metrics which quantify the probability of detection based on the noise models and collaboration among sensors. Let $c(p)$ denote the detection performance at point p. Since detection depends on resolution, we model $c(p)$ as proportional to the resolution at which a sensor s views p. This is a more accurate model than a purely distance based one, since the resolution depends not only on the distance but also the zoom setting of the camera and obstacles in the medium. Methods to obtain information about obstacles in the medium were discussed in [31]. For simplicity we assume no coordination among sensors for detection, and thus $c(p)$ is based on the nearest sensor observing p.

To aggregate the above factors into a single metric, consider a linear combination of the two metrics, $f(p)$:

$$f(p) = w * c(p) + (1-w)\frac{1}{\tau(p)} \tag{2}$$

The choice of the weighting parameter w, where $0 \leq w \leq 1$, determines the proportion of the contribution of delay and detection terms in the metric.

To bias the network configuration to regions where more events are expected, we further weight the above metric by the event distribution over the region covered. Suppose this distribution is known at all points $p \in A$ and is denoted $q(p)$. Then, define the utility metric at point p as:

$$u(p) = f(p) * q(p) \tag{3}$$

Suppose the pose of a sensor node i is denoted by $\mathbf{s}_i = \{\theta_p, z\}$, where θ_p and z represent the pan and zoom settings respectively. The network configuration, for a network of N nodes is then denoted by $\mathbf{S} = \{\mathbf{s}_1, \mathbf{s}_2, ..., \mathbf{s}_N\}$.

Using the utility metric defined above for each point, the overall utility of a network configuration, $\mathbf{S}$, is:

$$U(\mathbf{S}) = \sum_{p \in A} u(p) \tag{4}$$

Given the pan and zoom ranges of a node, the node state $\mathbf{s}_i$ lies in a set $S_i = \{[\theta_p^{min}, \theta_p^{max}], [1, z^{max}]\}$, where the superscripts *min* and *max* represent the minimum and maximum possible values respectively for the superscripted parameters. Let S represent the resultant set of possible network configurations. It is given by:

$$S = S_1 \times S_2 \times \ldots \times S_N \tag{5}$$

Choosing a good network configuration thus becomes an optimization problem:

$$\mathbf{S}_{opt} = \max_{\mathbf{S} \in S} U(\mathbf{S}) \tag{6}$$

This problem is NP-hard, as shown in [32], leading to high computation complexity. Also, the communication overhead of solving it centrally will be high if the number of cameras is large. We thus present a distributed algorithm for the above optimization.

Consider the utility function $U(\mathbf{S})$ defined in equation (4). This function is inherently distributed in that the utility value at a point p can only be influenced by a subset of the sensors: the ones which have p in their line of sight and within the maximum range of view from one or more of their possible poses. Denote the set of sensors which influence the utility value at p by $\beta(p)$. Collect together into a sub-region B_j, all those points which share the common influencing sensor set $\beta(p)$, and denote this sensor subset by β_j. Suppose the entire region A can be divided into K such sub-regions. By definition, the $\{B_j\}_{j=1}^{K}$ are disjoint (the corresponding β_j are not). The utility function can now be decomposed into a summation over these disjoint sub-regions:

$$U(\mathbf{S}) = \sum_{j=1}^{K} \sum_{p \in B_j} u(p) \tag{7}$$

Since each B_j corresponds to a sensor subset β_j and a set of N sensors may have 2^N subsets, the number of terms in the above summation is potentially exponential. We can however map the above decomposition to one where the number of terms is linear in N. Let γ_i denote the subset of sensors which is the union of all subsets β_j to which sensor i belongs:

$$\gamma_i = \{j | \exists \beta_k \quad \text{such} \quad \text{that} \quad (i,j) \in \beta_k\} \tag{8}$$

The utility in the region within range of sensor i can only be affected by the sensors in γ_i. Given the state of each sensor in γ_i, sensor i can compute the set of points Γ_i at which i gives the highest $u(p)$ value. The sub-regions Γ_i are disjoint (though the sensor subsets γ_i are not) and $\{\Gamma_i\}_{i=1}^{N} \cup B_K = A$. Hence the utility function can be decomposed as:

$$U(\mathbf{S}) = \sum_{i=1}^{N} \sum_{p \in \Gamma_i} u(p) \tag{9}$$

The interesting property of this decomposition is that each each sensor only needs to communicate with sensors in γ_i to determine $U(\mathbf{S})$ over Γ_i. We call γ_i the neighborhood of sensor i. The set γ_i includes all sensors which may affect the computation of Γ_i from any pose selected at those sensors. Suppose the region influenced by sensor i from all its poses is V_i. Then, the set γ_i includes potentially all sensors influencing any point in V_i. The expected cardinality of this set depends on the deployment density and the maximum distance up to which a sensor may influence coverage. For most practical sensors, the distance up to which they may sense is bounded, which implies that the cardinality of γ_i stays constant even as the network size, N, increases, at a given deployment density. We now describe a distributed algorithm which requires each sensor to coordinate only with sensors within γ_i and yet optimize the global utility.

Suppose a mechanism is available to ensure that when sensor i is changing its pose, all other sensors in γ_i will remain static. The communication protocol to realize this mechanism, and other implementation details will be discussed in section 3.3.2. The algorithm then proceeds as follows.

Sensor Pose Update: Sensor i searches over all its possible pan settings, and chooses the one for which the utility evaluated over V_i is maximum:

$$\theta_p(i) = \max_{\theta_p \in [\theta_p^{min}, \theta_p^{max}]} \sum_{p \in V_i} u(p) \tag{10}$$

This is a one dimensional search and thus computationally tractable. Computation may be further simplified by noting that the utility over the entire V_i only needs to be calculated once and then it may be updated for the changes in covered region due to a pose change. After selecting its best pan setting, the sensor similarly selects its best zoom setting. After this pose update, sensor i stops and some other sensor j waiting on i may update its pose, again making sure that sensors in γ_j stay static during its update step. At each step the algorithm is searching in one dimension of the 2N-dimensional search space; hence we refer to this algorithm as Incremental Line Search (ILS).

This change in pose at i affects all Γ_j for $j \in \gamma_i$. However, it does not affect sensors not in γ_j and they may be changing their poses in parallel with sensor i using the same method. This property makes the algorithm distributed over neighborhoods γ_i. The sensors update their poses in an arbitrary order.

3.3.2 Network Protocol

We now describe a motion coordination protocol to implement the distributed optimization algorithm described above.

The first step is the determination of a mechanism to ensure that for each node i changing its pose, the set of nodes in γ_i stay static, and the number of nodes which can update their pose at any given time is maximized. This can be viewed as a graph coloring problem where for a given node i, all nodes in γ_i are treated to be adjacent to it. Each node must be allotted a color such that no two adjacent nodes have the same color while the number of colors is minimized. With this, nodes of one color can update their poses simultaneously and the number of iterations required to allow each node update once would be equal to the number of colors used. Efficient graph coloring heuristics are known and hence if the entire network graph is known at a central location, any such heuristic may be used. However, we do not assume that a central view of the network graph is known and propose the following heuristic for achieving a graph coloring in a distributed manner. This procedure is inspired from common RTS-CTS based wireless MAC protocols for avoiding interference, though we do not need the communication channel to be wireless for its execution. The protocol at each node i is described in Fig.6.

The protocol requires a random wait before sending a Request to Update (RTU) packet in order to ensure that over a long duration, most nodes will get a change to update their poses rather than a just few well positioned nodes with smaller number of neighbors repeatedly winning the contention. No sequential ordering within a neighborhood is imposed, and it is not required that each node get equal number of chances to update its pose.

The protocol exploits medium information about obstacles; such information may be obtained using methods presented in [31]. If not available, line of sight calculations will not account for the occlusions, but this is no worse than another sensor network not using ILS and not aware of its detection performance. The protocol also uses the event distribution $q(p)$ when available, such as obtained and updated using methods detailed in [32]. If the event distribution cannot be learned, a uniform distribution is assumed and we compare the effect on performance due to lack of the event distribution information in section 4.1.

The termination condition in the protocol is chosen for ease of distributed implementation. In centralized optimizations, a stopping criterion that can determine when to stop the iterations is to check when the change in the estimate of the optima has become insignificant over the past few iterations. However, in the distributed setting, the global utility metric is not available to any node. Even if the parameters in control of a particular node are not changing from one iteration to the next, the parameters at other nodes may be changing and determining the stopping time would thus require global coordination. To avoid this global coordination, the termination condition proposed in Fig. 6 allows each node to stop based only on local information. The value of Δ is set to 0.1% of the previously measured utility over V_i and may vary from sensor to sensor. No node may know when the entire network has reached a stable state. Each node may stop its motion and restart it multiple times before reaching a stable condition. The convergence properties of the ILS algorithm hold with this distributed termination.

As an example, consider the coverage with four cameras shown in Fig. 7(a). The black regions are obstacles that block line of sight and the white regions are uncovered. The shaded regions represent areas covered by each camera, where the shade corresponds to the utility metric at the point due to the camera best covering it. A darker shade (more red shade, in colored view) represents higher utility. This configuration is just a regular tessellation of the four cameras placed to cover the square space. Fig. 7(b) shows the

1. **Contention:** Wait for a random back-off period, and send Request To Update (RTU) to each node in γ_i. Wait for $T_{timeout}$. If no CONFLICT message is received, begin updating as per step 2. Each node which receives an RTU packet, refrains from sending an RTU until it receives an Update Finished (UF) packet from each node which had sent it an RTU. If a CONFLICT message is received, wait for UF message from the sender of CONFLICT message, and retry the RTU after random back-off. This ensures that within γ_i, only one node moves at any instant.
2. **Update:** If an RTU is received anytime during this phase, send a CONFLICT message to the sender of the RTU.
   (a) **Medium and Phenomenon Information:** Use the medium mapping service to get information about obstacles within V_i, if available, to use in the calculation of $c(p)$. Retrieve the $q(p)$ over the region V_i as acquired locally in the phenomenon learning phase.
   (b) **Neighbor Pose Information:** Look up the list of UF packets received up to now. UF packets contain the sender's chosen {pan, zoom} pose after an update. Thus the poses of all nodes within γ_i which have updated up to now are known. Using these and the medium information, the utility metric over V_i can be calculated.
   (c) **Pose Selection:** Select the pan and zoom settings as per the ILS algorithm. Transmit packet UF={identity, chosen pose}. Other nodes may now contend to update their poses. After a node receives a UF packet from every node from which it had received an RTU, it goes to Step 1 unless the termination condition (Step 3) is met.
3. **Termination:** When a node discovers that the pose computed using the ILS algorithm does not change the utility over V_i by more than a threshold Δ compared to its current pose, it need not update its pose for that iteration. Each node maintains the past two UF packets received from its neighbors. Now, if the poses of all the neighbors have remained constant in the past two UF packets received, the node no longer needs to update its pose. If it has itself transmitted at least two UF packets, it terminates its motion algorithm until such time as another neighbor updates its pose or the information regarding $q(p)$ or the medium changes.

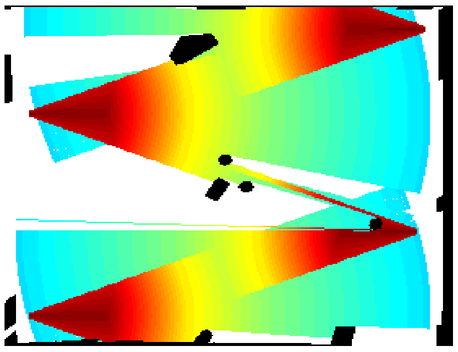
Figure 6. Distributed motion coordination protocol for optimizing network configuration.

network configuration optimized by ILS. Note that the zoom and pan settings of the cameras have changed. This not only increases the area covered and hence the detection performance but also, the actuation delay for many regions is reduced as the increased zoom means a smaller zoom change is required for the actual sensing task.

The proof of correctness and convergence of the coordination algorithm as realized via the above network protocol is given in [32]. The convergence is guaranteed for the realistic utility metric developed above without linearity or differentiability assumptions, for the distributed termination condition described above, and without any assumption on the ordering of actuation steps during the distributed operation.

In addition to convergence to a good state, an additional properties of interest for this reconfiguration algorithm is its convergence speed. Also, since the optimization problem solved is NP-hard, the solution found by the distributed algorithm is only an approximation to the optimal one. The proximity of the converged state to the global optimum is also an important performance metric. Both these characteristics are studied through simulations with a large number of cameras in [32].

Let us compare the potential advantage of coordination among cameras using ILS as opposed to no coordination. When no coordination is used, each camera may optimize its pose independently of others, and it need not know its own or any other camera's location coordinates. We evaluate this in simulation with a network of 25 cameras randomly located with obstacles laced within the deployment area, thus creating a realistic setting. Fig. 8 shows the utility metric achieved for the configuration after the proactive phase of actuation. The white bars shows the utility metric achieved through coordination using ILS algorithm, plotted for 10 different runs. Since the order in which cameras actu-


(a) Original configuration

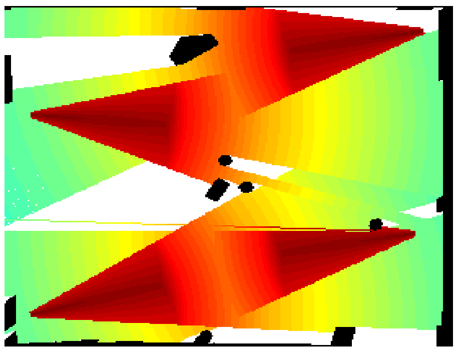

(b) Optimized configuration

Figure 7. Illustration of ILS operation.

ate is random during ILS actuation, the final utility varies for each run. The black bars show the utility of the configuration achieved when each camera has optimized its pan, tilt, and zoom pose independently. This is not varying but re-plotted with each random run of ILS for comparison. The utility achieved with coordination is significantly greater than that without coordination.

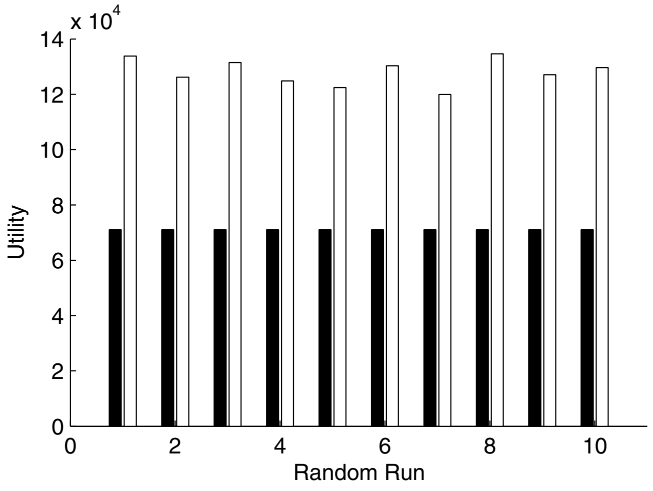


Figure 8. Advantage due to coordination.

The proactive reconfiguration method presented above helps minimize the actuation delay required in addition to maintaining coverage for detection, using small motion.

3.4 Applicability

3.4.1 Technology Context

While our prototype is implemented using the current technology generation of cameras, we believe that our system architecture is designed to support the anticipated technology trends in this domain. More compact and lower power cameras are becoming available for use in wireless sensor networks. MEMS based motion technology will enable significantly more compact and lower power pan and tilt mechanisms for such devices. Electrically controlled lens focus for changing the zoom, such as being developed for zoom-capable cellphone cameras, will enable the zoom capability to be accessible in compact low power wireless camera nodes likely to be used in future sensor networks. Non-mechanical pan-tilt-zoom cameras that use higher resolution image sensors enabled by silicon advances but only transmit a portion of the sensed image due to bandwidth constraints are also emerging. As more and more applications that use image data emerge given the increasing accessibility of the underlying technology, our methods for increasing the resolution will be increasingly relevant for newer generations of camera networks. Further the potential for significantly larger numbers of cameras made feasible through these emerging technologies will directly benefit from the distributed and scalable nature of our methods.

Also, as technology evolves, the image resolution in terms of number of pixels at a camera and the processing capabilities may increase, such as with Moore's Law. This may enable a given number of static cameras in a network to achieve a higher resolution than possible with the current generation hardware. For certain applications, this increased resolution may mean that the required resolution becomes feasible with static cameras alone. The architecture we presented may be used to obtain still higher resolution when it helps enable newer applications or improve the performance of existing ones further.

3.4.2 Relevance to Sensors Other Than cameras

Motion is useful beyond cameras or line of sight sensors as well. We mention an example where constrained motion helps increase the resolution for point sensors. Consider an application where the concentration of nitrate radical (NO_3^-) is to be mapped across the cross-section of a river, such as shown in Fig. 9. The available nitrate sensor node has a

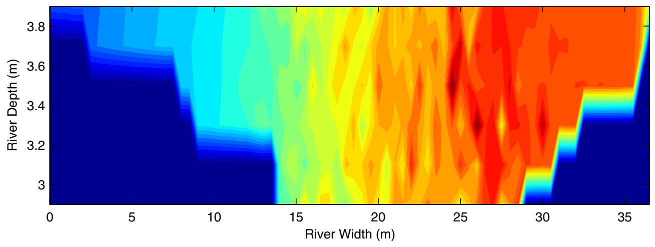


Figure 9. Nitrate concentration (mg/l) across the cross-section of a river.

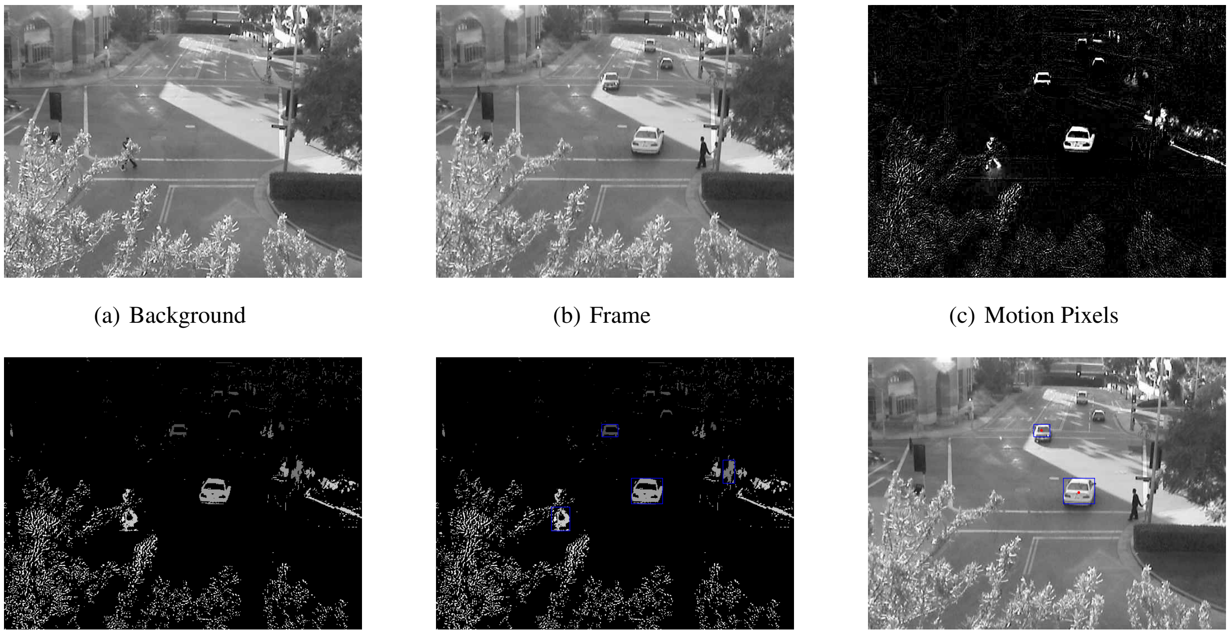
transducer surface $0.1m \times 0.1m$ that is used for sensing. Suppose nitrate concentration is to be mapped at 0.5m resolution. Then, installing the available sensor at every 0.5m across the river cross-section along with the deployment infrastructure such as metal trusses to support the sensors in the river is likely to interfere with the nitrate concentration map. On the other hand, a sensor with constrained motion supported via a cable, such as developed in [33], that can move in a 2D plane across the river cross-section, can map the nitrate concentration at the required resolution without interfering significantly with the phenomenon. Assuming the concentration map does not change within the time it takes for the sensor movement to be completed, the nitrate concentration can be mapped at the required resolution[4].

4 Implementation

We use off the shelf PTZ cameras [30] for our prototype. The motion capabilities of this camera have already been tabulated (Table 1). Four of these cameras are mounted on a building patio on one side of the scene to be photographed, shown in Fig. 1(a). Any available networking technology that has sufficient bandwidth to support video data, may be used and we used Ethernet. The cameras also allow for a WiFi card to be added in its PCMCIA slot and that may be used if a wireless connection is preferred. Since we did not have access to change the firmware on the camera, we developed our software on an external processing platform but the methods could well be implemented on the camera processor itself in actual production.

Our objective is to evaluate the proactive motion coordination methods developed as part of our system architecture.

[4]The latter method was in fact used for generating the nitrate concentration map shown in Fig. 9 in [33].


(a) Background (b) Frame (c) Motion Pixels

(d) Blobs (e) Aspect Ratio (f) Filtered Events

Figure 10. Detecting regions of interest using motion detection.

The key parameter of interest here is the actuation delay required for actuating to any event detected. Since the reactive methods required to carry out the actuation step in response to a detected event are not developed as part of our work, we assume a simple method where each camera will zoom in to an event detected in its field of view. Issues in the reactive phase, such as scheduling the camera for multiple simultaneous events [18] or tracking an event as it moves through the covered area [17, 19] are not being studied here. We will thus compare the network configurations achieved by our method and other methods by measuring the actuation delay per event averaged over a large number of events.

To obtain a realistic trace of event locations in the covered area of interest, we used motion detection on a live stream of images coming from a camera mounted to view the scene shown in Fig. 1(a). The following data processing is performed to filter out events of interest, in this case, vehicles, in the images:

1. A background image is first captured and updated using an autoregressive filter as described in [34] to adapt it to gradual changes such as due to light variations in the outdoor scene (Fig. 10(a)).
2. Each frame captured by the camera is subtracted from this and the pixel values above a minimum threshold in the difference image are denoted motion pixels. A sample frame is shown in Fig. 10(b) and the difference image in Fig. 10(c). The shade in Fig. 10(c) corresponds to the magnitude of the difference value.
3. Adjacent motion pixels in this image are then clustered into groups, referred to as blobs. Each blob is shown in a different shade in Fig. 10(d). Blobs with pixel count below a minimum threshold are discarded, and a new shade is not used for such blobs in the figure. Several uninteresting motion events, such as tiny blobs corresponding to the edges of the foliage moving in the wind, are thus rejected.
4. The next step is to filter out the motion events of interest. A rectangle surrounding each blob with pixel count above the minimum threshold is calculated (Fig. 10(e)). The aspect ratio of each rectangle, i.e., the ratio of its height to the width is calculated. Blobs with aspect ratio, r, such that $0.2 \leq r \leq 1$ are assumed to be arising from vehicles. Blobs with an aspect ratio greater than 1 may arise due to pedestrians moving in the scene (height is greater than width) and blobs with an aspect ratio lower than 0.2 may arise due to noise blobs corresponding to edges that occur in the difference image due to small unavoidable camera shake.
5. The centroid of the larger blobs are calculated and assumed to be the location of the motion events corresponding to those blobs. The calculated centroids for the filtered blobs are shown superimposed on the captured frame in Fig. 10(f). These locations represent the events of interest where high resolution coverage is desired. We store these coordinates as the event coordinates corresponding to this frame for generating our event trace.

Note that we are not addressing the problem of detection method design in this paper and assume that the detection performance achieved by the available method, such as motion detection in our example, is sufficient for system operation. Also, the above procedure yields the location of the events in the image pixel coordinates. For our scenario where the plane of the road is known, these coordinate may be projected to the road surface. This may not be feasible in some applications and the true coordinates of the events in 3-space may be needed.

A total of 5000 events were detected using the above procedure in a live video stream. A histogram of these events over the scene coordinates is shown in Fig.11. Lighter shades correspond to regions with more events. The complete image dataset used is available through our CVS repository [3].

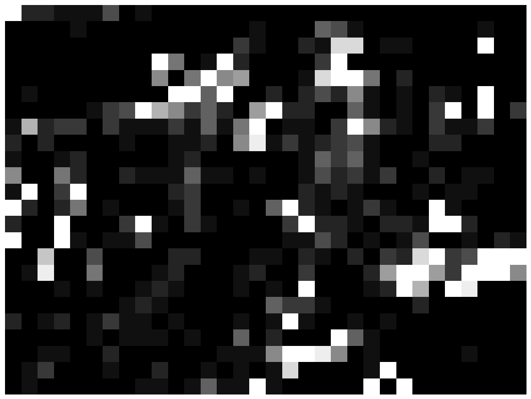


Figure 11. Event trace

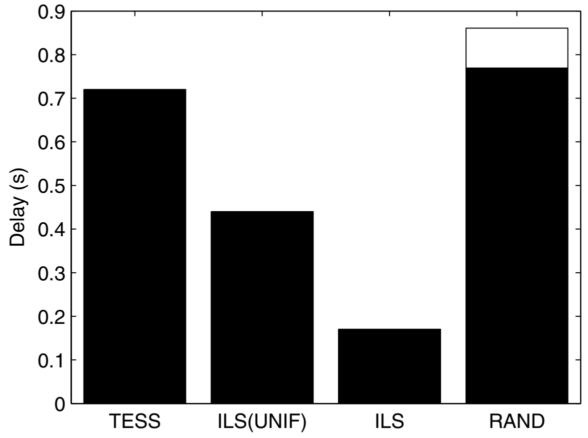
4.1 Evaluation

The proactive motion coordination methods are evaluated using this event trace and a set up of four cameras looking at the scene for which this event trace is collected. The actuation delay performance of a particular network configuration is calculated as follows: for each event in the trace, the time required by the camera having detection coverage (low resolution coverage) of that point to zoom in to provide the high resolution coverage is computed. This time varies with the extent of zoom, pan, and tilt motion required to reach that event. The computed time for each event is then averaged over all events.

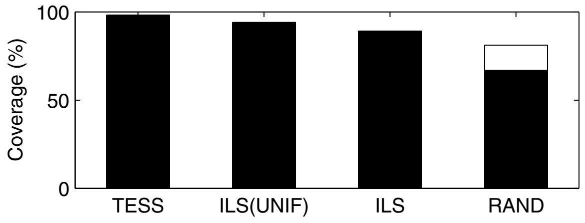
We compare four alternatives, labeled in the following graphs using the acronyms below:

1. TESS, a regular tessellation: The four cameras are arranged regularly with their zoom set to widest and each camera looking straight ahead.
2. ILS(UNIF): The ILS algorithm is used to optimize the configuration. Only 5X of the zoom is used since the remaining 20X is required for the reactive phase. Similarly, the excess pan range not required for the reactive phase is used. The tilt is ignored, but can be used in the same manner as pan.
3. ILS: The ILS algorithm is used with a knowledge of the true event distribution. The histogram generated from the event trace is used as the event distribution, and since this comes from the trace itself, it is a true event distribution. If the observed event distribution varies from distribution learned by the system, the delay performance may suffer somewhat; the exact deviation depends on the error in the learned distribution itself.
4. RAND: The cameras are configured randomly. Ten random configurations are tried and the average and standard deviation of the quantities being evaluated are plotted.

In the utility metric used for ILS, the tuning parameter w allows biasing the algorithm toward actuation delay or detection quality. Suppose we bias it to actuation delay, by using a higher weight $w = 0.9$ for the delay term and $(1-w) = 0.1$ for the detection term. The delay performance for the four



(a) Actuation delay



(b) Detection coverage

Figure 12. Performance of different network configurations, when optimization is weighted for actuation delay.

alternatives is plotted in Fig. 12(a). The corresponding detection performance, measured in terms of the fraction of the events detected is plotted in Fig. 12(b). The lighter shade bar segment on top of the bar for RAND is the standard deviation across the multiple random runs. Clearly ILS yields the best delay performance. The detection performance is comparable for all alternatives except RAND and is above 90%, showing that when ILS optimizes for delay, the penalty on detection is not significant.

Note that internally, the ILS algorithm is not using delay but the utility metric defined in section 3.3.1 for optimizing the configuration. This utility metric for the four alternatives is plotted in Fig. 13, to see how this internal measure relates to the observed application level performance metrics. The configurations obtained using ILS have higher utility. This is expected as the ILS algorithm changes the starting configuration to increase the utility metric. Among these, the utility is higher when the true event distribution is used.

On the other hand, we could bias the utility metric for detection performance, using the tuning parameter w provided by our algorithm. The results for this choice, i.e., the weight

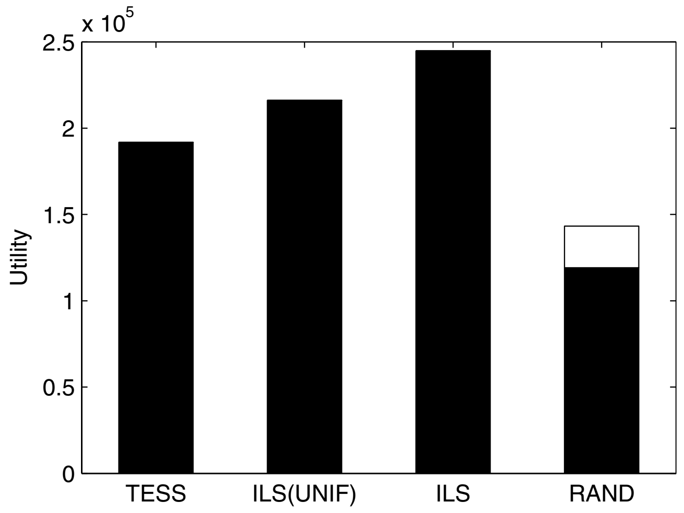


Figure 13. Utility metric evaluated for various configurations.

for delay term set to $w = 0.1$, and that for detection term set to $(1-w) = 0.9$, are shown in Figs. 14 and 15. As may be seen in Fig. 14(b), the detection performance has improved. Since the detection performance was already high, the improvement relative to Fig. 12(b) is small.

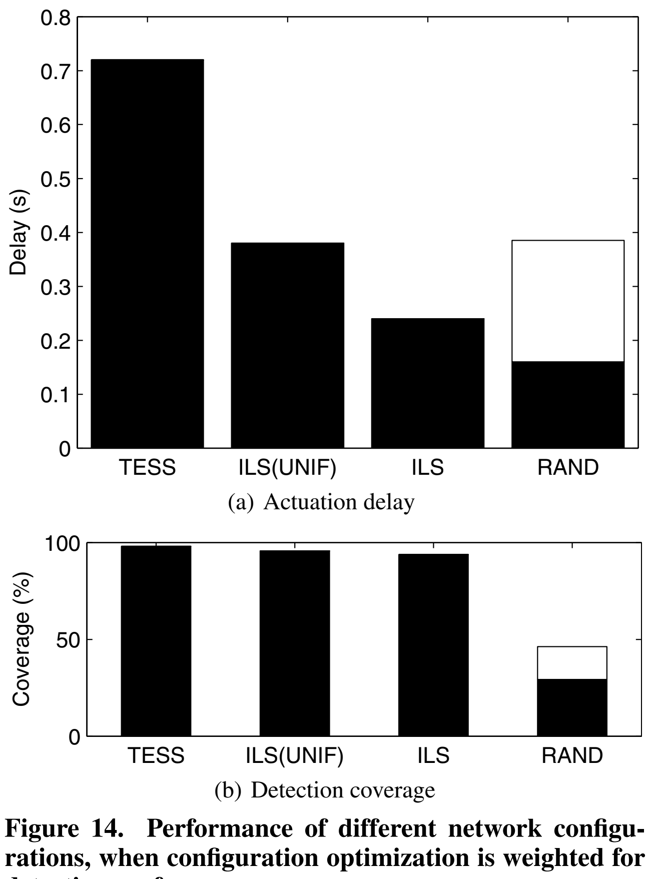


Figure 14. Performance of different network configurations, when configuration optimization is weighted for detection performance.

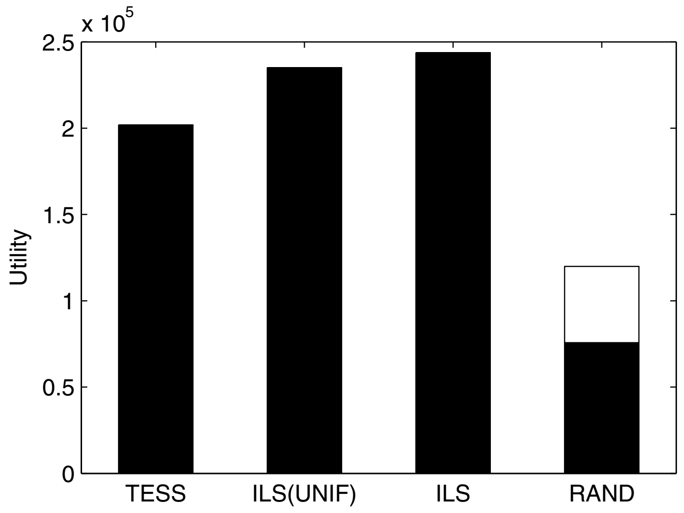


Figure 15. Utility metric for various configurations, when detection performance is weighted higher.

5 Conclusions

We discussed how controlled motion can help provide high resolution sensing in camera networks. While only limited motion capabilities with low resource overheads were used, the increase in resolution achieved was significant. The total sensing resource that may be deployed is often limited by deployment constraints arising from intrusion into the user space, the need for supporting installation, communication and power infrastructures, or even the system deployment and operating costs. For the given maximum sensing resource that may be used, our methods enable sensing at previously unachievable resolutions. We expect this increase in resolution to be beneficial to several applications; the extent of advantage to a particular application depends on the specific data processing methods employed.

The extent of resolution advantage depends on the tolerable trade-off in actuation delay. We presented methods which minimize the actuation delay for a required resolution. Our methods were evaluated using real world event traces collected using our system prototype. Significant advantage in actuation delay was achieved.

We believe that we have made a first attempt toward understanding the issues involved in using motion for virtual high resolution, and future research into this space may provide improved solutions. The proposed system architecture and reconfiguration methods were demonstrated for a one set of choices, motivated using a class of application scenarios similar to the example used in our prototype. Other systems may be designed for application scenarios with varying design requirements. Future work also includes integration of our proactive methods with motion control used in the reactive phase.

6 Acknowledgements

This research was supported in part by the National Science Foundation (NSF) under Grant Nos. ANI-00331481 and 0306408, and the Center for Embedded Networked Sensing (CENS) at University of California Los Angeles.

7 References

[1] E. Shechtman, Y. Caspi, and M. Irani. Space-time super-resolution. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(4):531–545, April 2005.

[2] Aman Kansal, William J Kaiser, Gregory J Pottie, and Mani B Srivastava. Actuation techniques for sensing uncertainty reduction. Technical Report 51, Center for Embedded Networked Sensing (CENS), University of California, Los Angeles, 2004.

[3] Aman Kansal. Coordinated actuation project, nesl cvs repository. http://cvs.nesl.ucla.edu/cvs/ viewcvs.cgi/CoordinatedActuation/ExperimentData/.

[4] K.-C. Wang and P. Ramananthan. Collaborative sensing using sensors of uncoordinated mobility. In *Proceedings of International Conference on Distributed Computing in Sensor Systems (DCOSS)*, pages 293–306, June 2005.

[5] A Howard, MJ Mataric, and GS Sukhatme. An incremental self-deployment algorithm for mobile sensor networks. *Autonomous Robots Journal*, 13(2):113–126, 2002.

[6] Jorge Cortes, Sonia Martinez, and Francesco Bullo. Analysis and design tools for distributed motion coordination. In *American Control Conference*, Portland, OR, June 2005.

[7] Reza Olfati-Saber and Richard M. Murray. Consensus problems in networks of agents with switching topology and time-delays. *IEEE Transactions on Automatic Control*, 49(9):1520–1533, September 2004.

[8] B. Grocholsky, A. Makarenko, T. Kaupp, and H.F. Durrant-Whyte. Scalable control of decentralised sensor platforms. In *Information Processing in Sensor Networks: 2nd Int Workshop*, pages 96–112, 2003.

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

[10] Ashley Stroupe. *Collaborative Execution of Exploration and Tracking Using Move Value Estimation for Robot Teams*. PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, September 2003.

[11] Vasek Chvatal. A combinatorial theorem in plane geometry. *Journal of Combinatorial Theory Series*, 18:39–41, 1975.

[12] Ugur Murat Erdem and Stan Sclaroff. Optimal placement of cameras in floorplans to satisfy task requirements and cost constraints. In *Omnivis2004, The fifth Workshop on Omnidirectional Vision, Camera Networks and Non-classical cameras*, Prague, May 2004.

[13] Xing Chen and James Davis. Camera placement considering occlusions for robust motion capture. Technical Report CS-TR-2000-07, Computer Science, Stanford University, 2000.

[14] Sameera Poduri and Gaurav S. Sukhatme. Constrained coverage for mobile sensor networks. In *IEEE International Conference on Robotics and Automation*, pages 165–172, New Orleans, LA, May 2004.

[15] Z Butler and D Rus. Event-based motion control for mobile sensor networks. *IEEE Pervasive Computing*, 2(4):34–42, October-December 2003.

[16] W. Merrill, L. Girod, J. Elson, K. Sohrabi, F. Newberg, and W. Kaiser. Autonomous position location in distributed, embedded, wireless systems. In *IEEE CAS Workshop on Wireless Communications and Networking*, Pasadena, CA, September 2002.

[17] Maurice Chu, James Reich, and Feng Zhao. Distributed attention for large video sensor networks. In *Intelligent Distributed Surveillance Systems 2004 seminar*, London, UK, February 2004.

[18] Cash Costello, Christopher Diehl, Amit Banerjee, and Hesky Fisher. Scheduling an active camera to observe people. In *ACM 2nd International Workshop on Video Surveillance and Sensor Networks*, New York City, NY, USA, October 2004.

[19] R Collins, A Lipton, H Fujiyoshi, and T Kanade. Algorithms for cooperative multisensor surveillance. *Proceedings of the IEEE*, 89(10):1456 – 1477, October 2001.

[20] M. Trivedi, K. Huang, and I. Mikic. Intelligent environments and active camera networks. In *Proc IEEE Intl' Conf. systems, man, and Cybernetics*, volume 2, pages 804–809, 2000.

[21] S. Stillman, R. Tanawongsuwan, and I. Essa. A system for tracking multiple people with multiple cameras. Technical Report GIT-GVU-98-25, Georgia Institute of Technology, Graphics, Visualization, and Usability center, 1998.

[22] Mohammad H. Rahimi, Rick Baer, Obimdinachi I. Iroezi, Juan C. García, Jay Warrior, Deborah Estrin, and Mani B. Srivastava. Cyclops: in situ image sensing and interpretation in wireless sensor networks. In *SenSys*, pages 192–204, 2005.

[23] A. Savvides, C.C. Han, and M.B. Srivastava. Dynamic fine grained localization in ad-hoc sensor networks. In *Proceedings of the Fifth International Conference on Mobile Computing and Networking (Mobicom)*, pages 166–179, Rome, Italy, July 2001.

[24] D. Moore, J. Leonard, D. Rus, and S. Teller. Robust distributed network localization with noisy range measurements. In *Proceedings of the Second ACM Conference on Embedded Networked Sensor Systems (SenSys)*, page 5061, Baltimore, MD, November 2004.

[25] D. Devarajan and R. Radke. Distributed metric calibration for largescale camera networks. In *First Workshop on Broadband Advanced Sensor Networks (BASENETS), in conjunction with BroadNets*, San Jose, October 2004.

[26] Stanislav Funiak, Carlos Guestrin, Mark Paskin, and Rahul Sukthankar. Distributed localization of networked cameras. In *Inormation Processing in Sensor Networks*, Nashville, TN, USA, April 2006.

[27] B. Batagelj, F. Solina, and P. Peer. 15 seconds of fame - an interactive, computer-vision based art installation. In *Proceedings of the 12th ACM International Conference on Multimedia*, pages 764–765, New York, NY, USA, October 2004.

[28] H. Wang, J. Elson, L. Girod, D. Estrin, and K. Yao. Target classification and localization in a habitat monitoring application. In *Proceedings of the IEEE ICASSP 2003*, Hong Kong, April 2003.

[29] M. Maroti, G. Simon, A Ledeczi, and J. Sztipanovits. Shooter localization in urban terrain. *IEEE Computer Magazine*, August 2004.

[30] Sony SNC-RZ30N data sheet. http://bssc.sel.sony.com/, 2004.

[31] A. Kansal, J. Carwana, W.J. Kaiser, and M.B. Srivastava. Acquiring medium models for sensing performance estimation. In *IEEE SECON*, September 2005.

[32] A. Kansal, W.J. Kaiser, G.J. Pottie, and M.B. Srivastava. Reconfiguration methods for mobile sensor networks. Technical Report TR-UCLA-NESL-200601-03, Networked and Embedded Systems Laboratory (NESL), University of California, Los Angeles, January 2006.

[33] T.C. Harmon, R.F. Ambrose, R.M. Gilbert, J.C. Fisher, M. Stealey, and W.J. Kaiser. High resolution river hydraulic and water quality characterization using rapidly deployable networked infomechanical systems (NIMS RD). Technical Report 60, Center for Embedded Networked Sensing, UCLA, February 30 2006.

[34] M. Saptharishi, C. Diehl, K. Bhat, J. Dolan, and P. Khosla. Cyberscout: Distributed agents for autonomous reconnaissance and surveillance. In *Mechatronics and Machine Vision 2000*, pages 93–100, September 2000.