Hindawi Publishing Corporation
EURASIP Journal on Image and Video Processing
Volume 2010, Article ID 865803, 13 pages
doi:10.1155/2010/865803

Research Article

Design and Optimization of the VideoWeb Wireless Camera Network

Hoang Thanh Nguyen, Bir Bhanu, Ankit Patel, and Ramiro Diaz

Center for Research in Intelligent Systems, University of California, Riverside, CA 92521, USA

Correspondence should be addressed to Hoang Thanh Nguyen, nthoang@cs.ucr.edu

Received 2 January 2010; Accepted 16 August 2010

Academic Editor: Peter Tu

Sensor networks have been a very active area of research in recent years. However, most of the sensors used in the development of these networks have been local and nonimaging sensors such as acoustics, seismic, vibration, temperature, humidity. The emerging development of video sensor networks poses its own set of unique challenges, including high-bandwidth and low latency requirements for real-time processing and control. This paper presents a systematic approach by detailing the design, implementation, and evaluation of a large-scale wireless camera network, suitable for a variety of practical real-time applications. We take into consideration issues related to hardware, software, control, architecture, network connectivity, performance evaluation, and data-processing strategies for the network. We also perform multiobjective optimization on settings such as video resolution and compression quality to provide insight into the performance trade-offs when configuring such a network and present lessons learned in the building and daily usage of the network.

1. Introduction

We describe the design and development of a new laboratory called VideoWeb to facilitate research in processing and understanding video in a wireless environment. While research into large-scale sensor networks has been carried out for various applications, the idea of massive video sensor networks consisting of cameras connected over a wireless network is largely new and relatively unexplored. The VideoWeb laboratory entails constructing a robust network architecture for a large number of components, including cameras, wireless routers and bridges, and video-processing servers. Hardware and equipment selection needs to take into account a number of factors, including durability, performance, and cost. In addition, VideoWeb requires a number of software applications including those for data recording, video analysis, camera control, event recognition, anomaly detection, and an integrated user interface.

Challenges for the design of VideoWeb include creating a wireless network robust enough to simultaneously support dozens of high-bandwidth video cameras at their peak performance, providing power and connectivity to cameras, building a server farm capable of processing all the streaming data in realtime, implementing a low-latency control structure for camera and server control, and designing algorithms capable of realtime processing of video data.

The paper is organized as follows. In Section 2, we cover related work and contributions of this paper. Section 3 discusses the requirements and specifications used in designing the system and discusses the technical challenges and solutions for actual implementation. Section 4 delves into the performance metrics used to evaluate the system. Section 5 concludes with closing comments and lessons.

2. Related Work and Contributions

Many wireless camera platforms have been proposed [1–3], and emerging research in the design of wireless camera networks includes those with customized camera hardware nodes (e.g., CITRIC [4], eCAM [5]) including iMote2 and WiCa-based networks [6, 7], as well as networks with carefully-calibrated cameras [8].

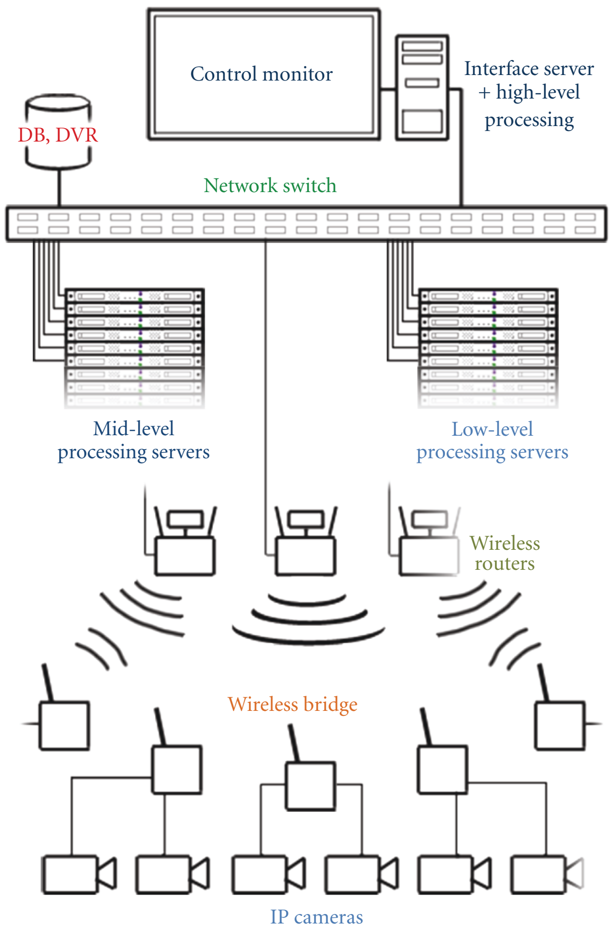

Control monitor
Interface server + high-level processing
DB, DVR
Network switch
Mid-level processing servers
Low-level processing servers
Wireless routers
Wireless bridge
IP cameras

Figure 1: Overall architecture. Top down: a single interface is used for direct control of any server/camera and high-level processing (e.g., user-defined face recognition). The server connects to a switch which hosts a database and joins two sets of servers: a series of mid-level (e.g., feature extraction) and low-level processors (e.g., detecting moving objects). The switch connects to routers which communicate with wireless bridges connected to the IP cameras.

This paper makes the following contributions.

(1) We expand upon [9] by exhaustively detailing the design considerations made in building the VideoWeb wireless network in order to provide a general guideline for those looking to build their own network. We also discuss lessons learned from building and using the VideoWeb network so that others may benefit from our experience.

(2) We make the case for IP cameras and server-side processing by designing and implementing a system utilizing network cameras running on a softwarere-configurable server and network architecture. While we use conventional IP cameras without on-board camera processing, the configuration of the server-side processing is user-configurable and allows on-the-fly changes such as going from tiered-processing (e.g., low-level processing servers do object detection and send silhouettes to mid-level processors which generate object signatures and broadcast to high-level servers) to 1-to-1 camera-to-server processing which simulates the behavior of on-camera processing networks.

(3) We describe performance metrics on which to evaluate a video network's performance and show how multiobjective optimization can be used in order to discover Pareto-efficient settings for configuring the network.

3. Building a Camera Network

3.1. Choosing the Type of Network. There are many types of camera networks (e.g., wired versus wireless, multihop wireless, distributed versus central processing), but the most important factor in deciding what kind of network to build is determining the primary application. For instance, if a network's primary concern is surveillance (where reliability or maintaining uptime may be paramount), a hard-wired network may be the only way to satisfy said requirements. A wireless network, on the other hand, provides more freedom and allows cameras to go where hard-wired cameras cannot (restricted only by power source).

3.1.1. Our Requirements and Implementation. The VideoWeb network consists of a heterogeneous mixture of over 50 wireless pan/tilt/zoom (PTZ) network cameras, 3 mobile robots equipped with cameras (see Figure 5(c)), and a 128-core server rack for data processing. The network is designed to be a flexible general-purpose camera network for use as a research testbed for applications such as multicamera tracking, scene analysis, and 3D reconstruction, as well as for research in improving robustness of wireless camera systems. Our implementation utilizes wireless cameras in order to take advantage of the flexibility in camera placement and cost savings afforded by not having to run network cable through the walls and ceilings to connect each of the cameras.

The complete architecture of the VideoWeb network (Figure 1) is comprised of a camera component, a wireless component, an application server component (e.g., database servers, digital video recording servers), and a processing component comprised of 3 levels: a set of servers which process camera feeds at a low level (e.g., human detection, per-camera tracking), a set of servers which use this information for mid-level processing (e.g., feature extraction, multicamera tracking), and a master interface server which uses this data for high-level processing and user control (e.g., task assignment, scene analysis, face recognition). The high-level server is also used as an interface for the network. Using a central switch to connect the two levels of processing servers allows data to move flexibly across servers to minimize network latency.

The server architecture is designed as a 3-level tree hierarchy: a master high-level interface server communicates with a set of mid-level processing servers, which in turn process data received from a number of low-level servers. A server architecture physically connected in this fashion would entail cameras forwarding data to one set of servers which forward low-level data to another set of servers, and once more to the high-level server. To minimize network overhead, we use a central network switch to connect the servers (as opposed to physically tiering the servers with direct connections) and implement the server hierarchy in the network's DNS configuration and in the communication strategy of our software.

An interface server is employed to allow users to view live or processed data from the cameras and to manually assign processing tasks (such as running a particular algorithm on some arbitrary number of cameras) from a central location (for VideoWeb, this location is back in the laboratory away from the noisy server room). For the wireless component of the network, cameras and servers are bridged through a single-hop wireless network using wireless routers connected to the servers to communicate with wireless bridges located throughout the building which connect to the cameras.

The following sections detail the design considerations made in building the network.

3.2. Choosing the Right Camera. Choosing the wrong camera can be a costly mistake when building a large video network. When selecting a camera, a number of factors should be taken into consideration. Besides cost, these may include the following.

(i) Wired versus Wireless Cameras. Deciding between a wired or wireless camera is often a trade-off between whether or not speed and reliability can be sacrificed in order to gain flexibility and freedom in placement. Cameras which connect to a processing location (central or distributed server) with dedicated wire connections (e.g., Ethernet, audio/video cables) excel in providing improved speed and reliability. This comes at the cost of restricting installation locations to those which can be reached via physical cables and installation may prove to be very labor-intensive, expensive, or simply unfeasible. Wireless cameras on the other hand allow greater freedom in placement as well as offering the opportunity of mobility (in the case of non-stationary cameras, for example, robots, field sensors), but may sacrifice speed, reliability, and/or security. Cameras with built-in wireless are essentially stuck with the installed protocol (though 802.11n is also backwards-compatible with 802.11g). Since the IEEE 802.11n standard supercedes 802.11g, this tends to make 802.11g-dedicated cameras feel outdated (especially if streaming requirements later exceed the bandwidth of the protocol or find that frame rates suffer from congestion and the only way to improve the situation then is by installing larger antennas, routinely changing wireless channels, and/or installing wireless repeaters). Cameras with built-in 802.11n wireless are preferred over 802.11g in almost all cases due to the increase in bandwidth, range, and potentially a less-crowded frequency range (though this may change with time). It is worth noting that wired cameras which lack built-in wireless transmitters can easily be made wireless cameras via wireless bridges or adapters. This may be a better choice for long-lifespan camera networks which may need to be concerned with forward compatibility, for instance, as it avoids the network from being locked into any single standard. How easy it is to make this modification is affected by whether the cameras are digital or analog.

(ii) IP versus Analog CCTV. Digital versus analog in the context of video cameras is often an issue of convenience. Traditional analog closed-circuit TV (CCTV) systems are often simpler and more cost-efficient, but search and retrieval of data is cumbersome and any applications beyond surveillance and monitoring may be awkward or require dedicated systems for each application. IP systems, on the other hand, can be more costly and/or complex, but output digital streams easily processed on computers and can even be accessed anywhere in the world simply by putting them on an Internet-accessible connection. If the video streams will be subject to constant or routine processing, analysis, or retrieval, IP cameras offer greater convenience and all the benefits of cheap digital storage, but may require additional network and software training for those only familiar with traditional CCTV systems.

(iii) Single-Hop versus Multihop Wireless. If wireless cameras are to be used, there are two primary ways they can reach their processing/storage destination: via a single-hop connection (cameras connect directly to wireless router/receivers) or via multihop connections (cameras connect to other cameras and pass on data before reaching the router/receiver). Multi-hop networks impose additional complexity and hardware as well as increased latency, but gain flexibility and wireless coverage by essentially turning every camera into a repeater node; these are moresuited for cameras with on-board processing capabilities. Single-hop networks are recommended if it is viable (i.e., network routers can be installed in locations in which all cameras can reach) for purposes of lower latency and reduced hardware requirements.

(iv) External versus On-Camera Processing. Whether or not to perform processing on-camera or deferring processing to external computers/systems is impacted by camera capability/programmability and network latency and bandwidth. For instance, a multihop network may be too slow to permit active tracking if video needs to first be passed through several sensors before reaching a processor, whose control commands then need to be relayed across several more sensors before the camera ever receives the command to "pan left". Outside of basic scripting capabilities, most commercial cameras do not offer the flexibility or processing power to achieve processing tasks more complicated than basic motion detection or tracking. This issue often prompts network builders to develop custom programmable camera hardware for use in their systems [4–7]. On-camera processing can also reduce bandwidth consumption of the network (e.g, transmitting only areas of interest as opposed to full-frame video), while external processing allows a greater range of control and processing power.

(v) Pan/Tilt/Zoom (PTZ) versus Static Cameras. As the name implies, PTZ cameras offer active panning, tilting,

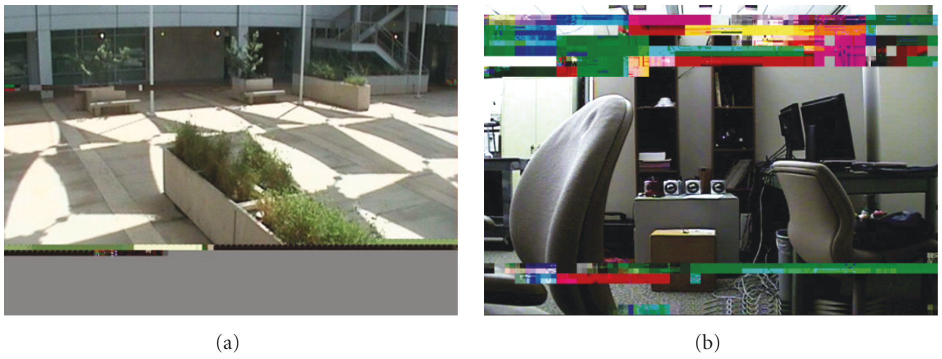


Figure 2: Stream corruption caused by network congestion may manifest in different ways depending on the video format. (a) corrupted Motion JPEG stream due to partial data, (b) corrupted MPEG-4 stream due to partial data.

Table 1: Camera behavior can vary radically across vendors and models. Under congested network conditions for example, cameras may permanently drop frames or attempt to resend missed frames at the expense of live data. The Panasonic camera in this case output "smoother" video (fewer frame drops between two successive frames) under heavy network congestion (until its on board cache is exhausted) at the cost of delays in upwards of 6 seconds.

	Panasonic WVNS202		Axis 215 PTZ	
Configuration	640 × 480 pixels, 0% compression		704 × 480 pixels, 0% compression	
Cameras per bridge	2	3	2	3
Frame delay (seconds)	<1.0	>**6.0**	<0.5	<1.0

and/or zooming capabilities whereas static cameras retain a permanent fixed field of view and orientation. PTZ cameras have the advantage of being able to cover larger areas (as a whole) and can zoom in or out to obtain better views of a scene as appropriate. This comes at the cost of increased complexity by requiring (manual or automated) control in order to take advantage of this capability. These cameras also contain moving parts, potentially affecting long-term maintenance. Static cameras on the other hand, are often less expensive and provide consistent scene coverage. In addition, they also often allow interchangeable lenses which can mimic some versatility of PTZ cameras by allowing one to customize a camera for certain applications, for example, installing a wide-angle lens to cover a larger area or installing a sharp telephoto lens to capture the entrance of a certain doorway (note that barrel distortion caused by using wider lenses should also be taken into account). Even with wider lenses, however, static cameras may require more installations to cover the same area as PTZ cameras and may do so with compromised quality (camera placement is often a balance between sacrificing area coverage for close-up detail).

(vi) Pan/Tilt/Zoom Speed and Magnification. If PTZ cameras are used, the responsiveness of such camera commands should be taken into consideration when choosing between models, as some cameras may respond or move too slowly to be useful for applications such as active tracking. Since the timing/latency specifications are often omitted by camera manufacturers, it is strongly recommended to experiment with trial cameras and testing if their PTZ speed is adequate before purchasing. In addition, the level of *optical* zoom may be important depending on the detail required for specific applications and the camera's physical distance from the scene. For most applications, digital zoom is worthless (at the raw capture stage) and should only be done in data processing.

(vii) Progressive versus Interlaced Cameras. All other things equal, progressive cameras should be chosen over interlaced cameras where possible. This may not aways be the case, however, as progressive models may offer reduced frame rate, resolution, or cost substantially more. While interlaced cameras can usally perform on-camera de-interlacing to avoid the combing artifacts inherent to interlaced video, such techniques tend to wash out fine detail for static objects and result in ghosting effects on moving objects ones (the alternative, processing only every other line in the video, also effectively halves the vertical resolution). There may be some exceptions to choosing a progressive camera, such as when a CMOS-sensor progressive camera has a rolling shutter which is so slow that its video exhibits noticeable skew on moving objects (also known as the "jello effect" as often seen in handheld cameras when the camera is panned too quickly), but even this may be preferred over the combing or ghosting artifacts from interlaced video.

(viii) Sensor Size and CMOS versus CCD. Sensor size is often more indicative of a camera's image quality than its stated resolution and this is true of video cameras as much as photographic cameras. Larger sensors tend to offer less image noise (especially in low light conditions) and sharper image quality. These sensors are typically either CMOS or CCD. While both sensors are used to achieve the same thing, complementary metal-oxide-semiconductor

(CMOS) sensors typically use a rolling shutter (i.e., light is captured in a sweep across the sensor) whereas charge-coupled device (CCD) sensors use global shutters (i.e., light is captured simultaneously across the sensor). The two are typically characterized by different kinds of artifacts each produces. For instance, CMOS sensors may suffer from skew on moving objects or scenes if its shutter speed is too slow, while CCD sensors are vulnerable to smearing artifacts when bright light sources overload a column or row of pixels. It is recommended to consider the typical environment the cameras will be used in (e.g., low light, indoor versus outdoor) and to trial all candidate cameras where possible.

(ix) Bandwidth: Video Format, Resolution, and Frame Rate. Resolution and frame rate go hand in hand as they will (in addition to video format) directly affect the bandwidth required for transmitting and storage required for archiving. Typical video cameras offer VGA resolution (640 × 480) at 30 frames per second, but newer high-definition (e.g., 720p or 1080p) cameras are becoming more readily available. While 640 × 480 resolution may be usable for many computer vision processing applications, those interested in face recognition (or better yet, face reconstruction) may find VGA to be particularly challenging to work with. Networks with particularly demanding requirements may want to consider specialty cameras, for example, super high-resolution cameras, hardwarestitched 360° cameras, or even high-speed cameras, though these tend to demand a premium. The output format of the camera will also affect image quality; in addition to the traditional and easy-to-decode Motion JPEG codec (essentially a large series of JPEG images concatenated together), many cameras also offer MPEG-4 output for reduced bandwidth and/or higher quality using the same bandwidth via interframe compression. Decoding the video for custom-built applications may be more difficult with MPEG-4 however, and video artifacts caused by stream corruption (e.g., network congestion, dropped packets) may appear less appealing (see Figure 2). With either format, we recommend using the open source libavcodec [10] library to facilitate decoding in custom applications.

(x) Power Requirements of Camera. Depending on the power requirements, cameras may be able to draw from existing power sources or require separate power supplies. Depending on the building or location, installing power cabling to the cameras may be easier than installing cabling for the data (in the case of a wired network) since a building's electrical architecture is usually more sophisticated than its network architecture. For the most remote installations which require more permanence than battery-operated sensors, readers may want to consider solar-powered wireless cameras.

(xi) Physical Appearance and Camera Enclosures. Appearance should not to be overlooked when it comes to installing cameras. If the cameras will be installed in an outdoor environment, large outdoor enclosures may invoke a sense of intimidation (see Figure 4(a)). It is recommended to take into consideration the environment the cameras will be installed to decide whether discreetness or visibility are higher concerns. As opposed to surface-mounting (installing a camera directly on a ceiling surface), flush-mounting (cutting a hole and installing a camera in the ceiling with the optics exposed) will provide a more discreet and streamlined appearance, but will require permanent alteration to the installation locations. If a network is temporary, readers are recommended to consider the life expectancy of the network before opting for flush mounting.

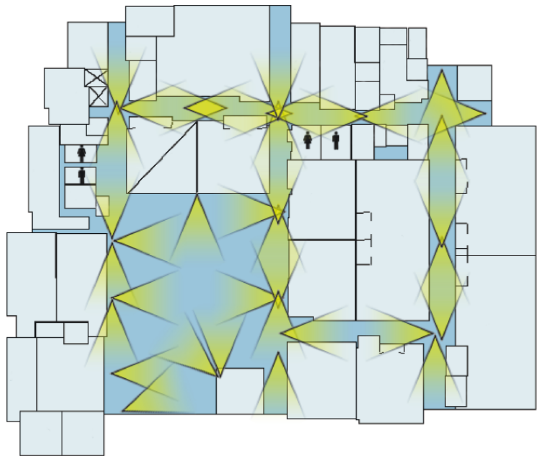


Figure 3: 37 camera locations cover the 14,300 square foot second floor of Engineering Building Unit II at the University of California, Riverside. Locations were manually selected and evaluated to ensure that usable fields of view were available for every square inch of the building from at least two viewpoints.

3.2.1. Our Requirements and Implementation. Initial specifications for the VideoWeb network required a minimum of VGA resolution (640 × 480 pixels) as well as a minimum of 20 frames per second as a threshold for acceptable realtime performance. In addition, we utilize digital IP cameras which provide a range of benefits such as streamlined processing (no digitizing required), relatively easy data storage, and simplified connectivity. The cameras are to be installed in an indoor and outdoor building environment which includes locations such as remote open spaces exposed to rain and corridors void of sunlight. As a camera network for long-term applications with year-round use, battery-powered cameras are not sufficient and we instead use network cameras with power adapters.

Among conventional pan/tilt/zoom (PTZ) cameras considered were the Panasonic WVNS202, Axis 214, and Axis 215 cameras. Besides cost, factors influencing camera choice include performance, physical size, and availability of non-intimidating outdoor enclosures. The Panasonic cameras were deemed unsuitable after experiments which showed that the video stream begins to lag when the network becomes congested (Table 1), that is, in the event of network throughput issues which limit cameras to low frame rates, instead of dropping frames, the Panasonic resends cached video frames stored in its buffer. The Axis cameras on the other hand, drops frames, maintaining a relevant video stream despite low frame rates. Between the two Axis cameras, the 215 was selected as the primary camera (despite being an interlaced camera from lack of availability at the time of selection) due to lower cost and lower mechanical latency when issuing PTZ commands.

Using 45-degree fields of view for the cameras, 37 locations were selected for complete coverage of the 14,300 square foot building (Figure 3). As such, the network consists of 37 outdoor cameras (36 Axis 215 PTZ cameras and a larger Axis 214 PTZ camera overlooking a courtyard) and 16 indoor legacy cameras. Camera locations are selected such that every square inch of the building is viewable by at least two cameras. In testing, each camera is capable of outputting a sustained 2.65 MB/second of Motion JPEG (M-JPEG) video at a peak of 30 frames per second when set to the maximum resolution of 704×480 pixels and a minimal compression setting of 0 (out of 100). This represents the maximum throughput and frame rate in an ideal environment (i.e., connecting to a camera via a direct ethernet connection and experiencing no frame drops). Other available resolutions of the cameras include 704×240, 352×240, and 176×120.

While the selected Axis 215 camera offers an outdoor dome enclosure, a dilemma was faced as there were no *discreet* outdoor enclosures for them; we had to find a way to make the cameras relatively weatherproof to withstand humidity and moisture. By choosing the Axis 215, we had to compensate for the lack of an available outdoor enclosure and improvise using the supplied flush-mount enclosures with smoked domes and surface-mount enclosures with clear domes, both designed for indoor installation. The solution was to use the surface-mount enclosures and make them weather-resistant by sealing the plastic seams with silicone sealant. In addition, the clear domes were interchanged with the smoked domes. The end result was a non-threatening camera dome suitable for surface-mounting at any of the 37 locations (see Figure 4(c)). Long-term effects of humidity, heat, and moisture on the cameras despite the sealed domes remains to be seen.

Electrical power was provided by installing dedicated power supplies in two of the building's electrical rooms and running conduit to the camera cluster locations where power outlets were installed. Since we had full control of the power by using our own power supplies, the cumbersome power adapters for the cameras were removed and the required power is supplied directly.

3.3. Choosing and Configuring the Network Hardware. The network hardware has a single purpose: to connect the cameras to the processing location(s) and to be as transparent as possible. Factors to consider when selecting network hardware include the following.

(i) For Wired Networking. If IP cameras are being used, it is recommended to install the highest-rated network cable available (Cat-6 ethernet cable as of this writing) which can still reach its destination (generally 100 meters for gigabit ethernet or 55 meters for 10-gigabit ethernet using Cat-6a). The cost difference may be marginal (over Cat-5/5e, for instance) while providing overhead in robustness in the event that newer higher-bandwidth cameras are installed to replace aging cameras. Ethernet extenders may be required if cable lengths exceed cable specifications.

(ii) For Wireless Networking: 802.11g versus 802.11n versus RF. If wireless IP cameras are used, it will likely be a choice between 802.11g and the newer 802.11n. If the choice is available (e.g., wireless bridges are being used to turn an ethernet camera into a wireless camera), 802.11n from our experience is a *major* upgrade from 802.11g for both increasing network throughput and signal strength. How much of an improvement may be influenced by congestion in the operating frequency range due to other wireless networks in the area. Determining a selection between analog RF transmitters, on the other hand, can be more difficult as the performance will vary more widely based on the power, frequency, and data being transmitted, as well as the environment. It is recommended to get a sample transmitter and to test each location cameras will be installed; this goes the same for wireless IP cameras, though wireless repeaters can be moreeasily installed to extend ranges. In addition, selected wireless routers should offer (at minimum) gigabit capabilities, especially if a large number of cameras are expected to connect to it.

(iii) Wireless Encryption. Use anything besides WEP [11].

3.3.1. Our Requirements and Implementation. Since many IP cameras (the Axis 214 and 215 included) do not have built-in wireless connectivity, a wireless bridge is required to provide this functionality. As such, the wireless bridges serve a single purpose: connect the cameras to the routers. Since the camera locations are often situated in clusters, it is desirable if the bridges can support multiple clients (i.e., have more than 1 ethernet port). This quickly narrows down the selection. A conventional IEEE 802.11g bridge made by Buffalo was selected due to its support of 4 ethernet clients; IEEE 802.11n bridges were only available in 1-port versions at the time of selection. This paper does not delve into the pros and cons of individual wireless protocols, though literature on this specific topic has been recently made available [12]. Performance testing on the Buffalo bridges revealed no outstanding issues, but prolonged testing showed that upgrading to 802.11n provides a worthwhile improvement for frame rates.

The wireless bridges were installed throughout the building in the ceilings and localized in clusters where possible to better facilitate maintenance and troubleshooting concerns (see Figure 4(b)). In total, 19 wireless bridges are used to provide connectivity for the 37 cameras. Though the bridges have 4 inputs, we only use 2; we do not take full advantage of the bridges' connectivity capabilities for a reason. We originally planned to optimistically use 3 cameras per bridge, but found 2 cameras (streaming simultaneously with maximum video settings) was the limit each bridge could support without experiencing heavy frame loss. The

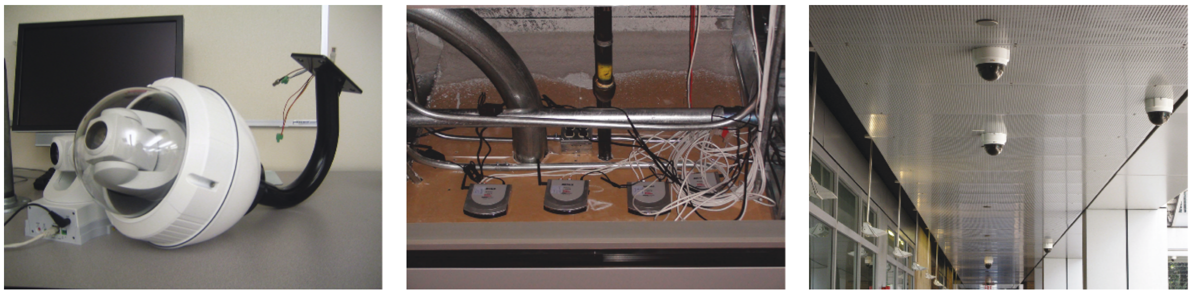


Figure 4: Installation of the cameras: (a) Axis 214 PTZ in an outdoor-rated enclosure; only one of these were installed (high above a large open courtyard) due to size and appearance), (b) wireless bridges installed in the ceilings to make the IP cameras wireless, (c) flush-mounted Axis 215 PTZ cameras in sealed indoor enclosures.

bridges are configured to communicate with the routers using WPA-PSK encryption.

At a maximum of 2.65 MB/s per camera (or 5.3 MB/s from each bridge), the network may be generating over 98 MB/s of data at peak performance. Gigabit routers are used to handle the amount of expected traffic and IEEE 802.11n capabilities are chosen to facilitate future upgrades. We use Linksys WRT350N routers for the first iteration of the network. Routers are split into two clusters receiving from two indoor locations. In total, 7 routers handle the traffic generated by the 19 bridges. The routers are configured to assign local addresses to the cameras and port forwarding is used to address the cameras from the servers.

3.4. Building the Server Hardware. Even with on-camera processing, it is still desirable to have external systems, either for data processing (due to much greater processing power) or storage. For digital networks, this system will likely be a number of computers. Whether specifying the hardware for these machines or building from scratch, it is useful to keep in mind a number of factors:

(i) Gigabit Network Connectivity. When dealing with streaming video data, always opt for (at minimum) gigabit network adapters. This is especially true if a single machine is expected to process multiple camera feeds. A gigabit network switch (or higher) is almost a requirement when connecting the servers together.

(ii) Hard Drives. For raw data processing, hard drive speed or capacity does not matter (all image-processing can be done from memory). For long-term storage, high-capacity hard drives in a redundant configuration (e.g., RAID 5) are recommended, though it is best to store these in a central high-density storage server (as opposed to distributed across several servers) in order to facilitate easier retrieval. Depending on the expected amount of constant incoming data, expensive high-RPM drives may or may not be necessary.

(iii) CPU. Depending on the multithreaded capabilities of the processing software (either your own or vendor-supplied software), multicore processors (and even multisocket motherboards) may provide a significant improvement in overall system performance. This is especially true if servers expect to process feeds from multiple cameras.

(iv) Memory. Images will be loaded into and read from memory constantly. Faster memory will reduce overhead, but *more* memory will likely only waste money as video images (processed on a per-frame basis) will not occupy very much space, even when uncompressed. There are exceptions, however, for example, when using super high-resolution cameras or for database applications which will cache large quantities of images (such as a face recognition database), so it is recommended to keep expandability in mind (i.e., 64-bit operating systems and motherboard memory capacity).

(v) Operating System. Though cross-platform code is preferred, the choice of operating system is determined mostly by the work/development environment the network operators and/or programmers feel most comfortable with. Server builders may want to take into account that most vendor-supplied software today is Windows-specific, however, but this may be irrelevant if you plan to develop your own processing software.

(vi) Server Location. If there are more than a few computers or servers in the system, it is recommended that they be moved to a dedicated server room with adequate cooling facilities; the heat, noise, and power consumption of all the servers can overwhelm most rooms.

(vii) Alternative Power. Uninterruptible power supplies (UPS) are recommended for all servers; their primary purpose is to allow the servers to gracefully shut down in the event of a power failure (or to buy time for backup generators to start up). This can be especially important for storage servers to help maintain the integrity of the servers' file systems.

3.4.1. Our Requirements and Implementation. We decided to go with a multicore system in order to enable more

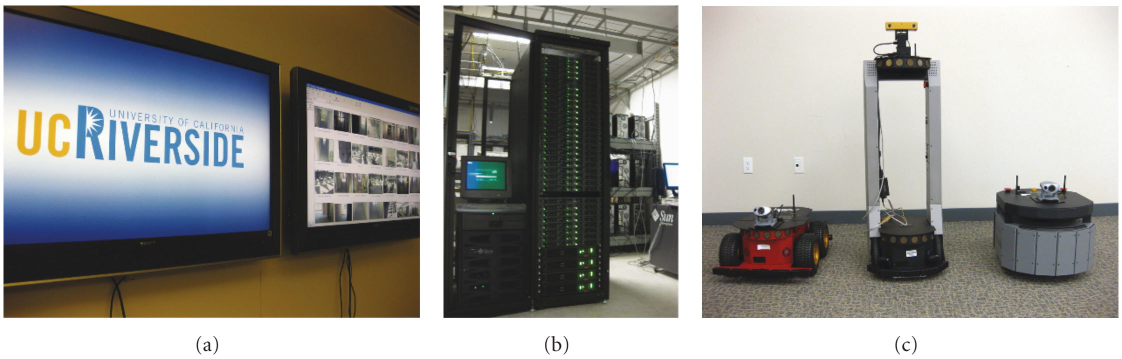


Figure 5: Processing hardware and mobile cameras: (a) control interface and monitors in laboratory, (b) 32-server processing backend connected to the interface, (c) cameras mounted on 3 robots add mobility to the network.

streamlined parallel data processing of multiple cameras per computer. Also, with our server architecture we have 3 levels of processing. If later on this amount of processing power is insufficient, each computer should have a second vacant CPU socket for another processor to allow doubling the processing power of the server farm if necessary without increasing the physical footprint of the system. For uniformity and to facilitate maintenance, all processing servers have the same hardware.

An idea to use conventional desktop computers for data processing was quickly discarded due to the difficulty in physically scaling ATX-sized desktop computers for a large number of cameras. Even using MicroATX cases would require a large amount of space to store the computers and would make moving the components/units particularly laborious and awkward. We instead opt for 1 height unit (1U) rack servers which can be housed in a single 42U rack enclosure with wheels for mobility.

In order to reduce contention over resources on the same machine from different camera processes, each processing server was specified with multicore CPUs and fast memory (Intel Core 2 Quad Q6600 2.4 GHz CPUs and 2 GB DDR2 800/PC 6400 memory). Though the Q6600 is not a true quad-core processor (2 dual-cores instead of 4 true cores), the support for additional threads is useful. Also, while we install 2 GB for our initial setup, we also use motherboards which are expandable to 24 GB of RAM for database applications in development. Gigabit ethernet cards are also selected to prevent any individual networking bottlenecks. Hard disks were given lower consideration, as most the processing nodes do mostly CPU processing and would not be storing data locally; the hard disks need only be sufficiently fast enough to run the operating system and RAM disks are setup in order to provide fast temporary storage for intermediate data. As such, conventional 80 GB SATA hard drives are used. Application servers such as database or recording servers, on the other hand may emphasize larger and faster hard disks.

Thirty two identical servers were built and installed into a server rack (Figure 5(b)) and then connected to an interface server with a pair of control monitors as an interface (Figure 5(a)). The building housing the servers fortunately has a suitable server room with adequate air conditioning and power connectivity. Electricity usage monitors were used to measure power consumption of the servers. The servers mentioned, for instance, peak at 198 W/1.65 A when starting up, use 132 W/1.14 A when idle, and consume 175 W/1.54 A under full load on all cores and hard drives. This data was then used to specify the uninterruptible power supplies (UPS) for the servers, which consist of four 2 U APC Smart-UPS 2200 VA/120 V batteries. Testing showed the batteries capable of supporting 8 servers each at full load for 5 minutes and 45 seconds, plenty of time to safely shut down (which can be configured automatically in software using UPS alerts) or to withstand short power outages.

3.5. Software System. In order to implement the tiered processing scheme of the servers, the software needs to be both clients and servers to facilitate the sending and receiving of video traffic and camera controls. Mid-level servers, for instance, may need to broadcast a stream of processed data to the high-level server for viewing by the user, while at the same time being able to download cropped object images from low-level servers.

The goal of the first software iteration was to control a networked camera using a customized program without the use of the supplied camera web interface or vendor-specific camera-management software included with most network cameras. One of the advantages of utilizing network cameras is that the camera control interface can be implemented through sending simple HTTP commands. To demonstrate this, a 10-line Python script was written for sending manual control commands to a camera. Once it was shown that it was easy to control the cameras, work started on a C++ application for the actual image processing.

3.5.1. Sample Program—Head Tracking. The basic algorithm framework used for head tracking was based on a gradient and color-based tracker [13] with additional tweaks. The first implementation of this was done in C++ and MATLAB [14]. Testing showed this program to be too slow for realtime processing, so a second iteration was written in pure C++ using OpenCV [15]. The tracker began with tracking synthetic object data consisting of randomly rotated rectangles of various sizes against a white backdrop. Once this stage was satisfactory, the next task was to grab live data from the camera.

Instead of relying on vendor-supplied software development kits (SDKs) which would have to be reintegrated into the processing software for potentially every type of camera, a generic camera controller was written. The SDK for the Axis cameras, for instance, relied on MFC-based [16] subroutines which would force development on Windows. In light of the amount of customization needed to incorporate a new SDK to do essentially the same things for different camera models, a generic cross-platform control framework was written from scratch. This control framework uses Boost.Asio [17] (a cross-platform socket wrapper) to directly send HTTP/1.1 [18] camera commands to a camera and uses the libavcodec library [10] to decode the streaming camera data. Using this approach, the software gains the benefit of being able to decode a large number of potential video streams and not just what a camera vendor has included with their SDK.

Networking communication between the three levels of servers is also implemented with Boost.Asio. For instance, processed results performed by the mid-level servers is compressed and broadcast as an M-JPEG stream, which is then parsed and displayed by the interface server.

4. Experiments for Performance Characterization and Optimization of the Video Network

4.1. Measurement Software. Software that comes with most IP cameras ranges from small camera control programs to full surveillance station applications. However, even the most expensive or sophisticated of these vendor applications can be unsuitable since they are usually targeted toward security applications and recording, playback, and camera control are often their sole function. Evaluating performance using these applications is subjective and raises the need for our own statistic-recording implementation.

A custom program was written to fulfill this function. Given an IP address and port number, the application proceeds to

(1) establish a connection with the camera

(2) attempt to download the M-JPEG video stream

(3) parse the stream into individual JPEG frames

(4) record realtime statistics about the stream.

The program records a number of statistics and measurements including bandwidth, shortest lag between two frames, and the average, minimum, and maximum amount of bandwidth required for each frame. The implementation is in C++ and uses the generic control framework written earlier.

4.2. Optimizing Camera Configuration. Depending on the task or application, there are numerous "optimal" ways to configure a network. For instance, maximizing video resolution and quality may be paramount for biometrics, particularly in face recognition where a large number of pixels on the face is beneficial to identifying features. Surveillance and alarm systems, on the other hand, may find reliability more important. For instance, it may be more important that every moment is recorded with minimal skipping (not only for evidence in the event of an incident, but also because security applications often employ vision-based motion detection). Object tracking in turn, may benefit most by sacrificing resolution in exchange for a high sustained frame rate.

Configuring the network may consist of changing camera parameters (e.g., resolution, compression) as well as physical network parameters (e.g., number of cameras per bridge, number of bridges per router, number of routers per square foot). The later is helpful in introducing a metric for minimizing labor and monetary cost. We define 5 metrics for measuring camera network performance, the first two of which are used as configuration parameters.

(1) *Resolution* (in pixels). This measures the size of each video frame in pixels (the higher, the better). This parameter consists of 4 levels on the Axis cameras (704×480, 704×240, 352×240, and 176×120).

(2) *Video Compression*. This parameter represents the amount of *lossy* video compression applied to the video by the camera. For M-JPEG streams on the Axis cameras, this represents JPEG compression and ranges from 0 to 100 (the lower, the better). In our experiments, we test 5 of these levels (0, 20, 30, 60, and 100).

(3) *Average Frame Rate* (in frames per second). This measures the number of *complete* frames received per second, averaged over the duration of a measurement trial (the higher, the better). The frame rate may range from 0 to a maximum frame rate of 30 on the Axis cameras.

(4) *Standard Deviation of Frame Rate*: This measures the consistency of the video. For instance, there may be two video streams both 20 frames per second each, but the first may output a constant 20 frames per second while the second video may be sporadic and go from 30 to 0 to 10, back to 30 and so forth (but still average to 20 in the end). This metric is useful in evaluating the stability of the video (the lower the deviation, the better) and is measured by recording the delay between every two frames (in seconds with millisecond resolution) and calculating the standard deviation.

(5) *Longest Lag Time between Two Complete Frames* (in milliseconds). This metric records the longest amount of time taken between any two consecutive frames (the lower, the better). This is insightful for evaluating a video stream's reliability (that is, it measures the longest amount of time a camera is "blind"). In addition to a depressed frame rate, this may be attributed to dropped/partial frames by the camera or data corruption/dropped packets undergone during transit.

4.3. Multiobjective Optimization Using Pareto Efficiency. We use the concept of Pareto efficiency to define which configuration of parameters is "better" than another. While this does not always tell a user which configuration should be used for a particular application, it serves to reduce the large number of possible configurations by showing which of those are usually "inferior"; a user only has to consider a configuration from the (potentially) much smaller Pareto set rather than every possible combination.

4.3.1. Inferiority and Noninferiority. Let M_1 be a vector of measurements of certain metrics for a camera and let M_2 be another trial of measurements on the same camera, but under a different parameter configuration. M_1 is said to be **inferior** to M_2 if and only if

(i) every measurement in M_2 is equal to or outperforms the corresponding measurement in M_1,

(ii) one or more measurements in M_2 outperform the corresponding measurements in M_1.

"Outperforms" is metric specific and means "greater than" or "less than" depending on how the metric is defined (e.g., a *higher* frame rate outperforms a *lower* frame rate and a *lower* lag outperforms a *longer* lag). M_2 is said to be superior to or *dominates* M_1 if M_1 is inferior to M_2. Finally, M_1 and M_2 are both said to be **non-inferior** if neither is superior nor inferior to one another.

In order for a measurement M_i to be **Pareto-efficient** (amongst a set), it must be non-inferior to every other measurement in that set. That is, it possesses at least one *advantage* over every other measurement when compared one-on-one (e.g., M_1 has higher frame rate against M_2, lower lag against $M_3, \ldots,$ higher resolution than M_n). The Pareto set is the set of all Pareto-efficient measurements and ideally, allows a user to discard a large percentage of inferior parameter configurations from consideration when setting the cameras.

4.3.2. Data Collection. Data collection consists of varying the resolution and compression parameters and recording the measurements from 37 cameras. In total, we iterate through 4 resolutions (704 × 480, 704 × 240, 352 × 240, and 176 × 120) and 5 levels of compression (0, 20, 30, 60, and 100) each. Five measurement trials are captured for each of the 37 cameras per configuration (100 trials total per camera). Each trial consists of streaming from the camera for 600 frames or up to 2 minutes (whichever comes first).

Camera footage is tested at 5 various points in the day across all cameras. This exposes the data to a variety of video footage ranging from bright open areas with upwards of 20 moving people in the scene, to dark and grainy footage of cameras monitoring lonely halls.

After data collection is completed, each camera is optimized individually to minimize camera, bridge, or router bias. This is done in $O(n^2)$ via exhaustive search (where n is the number of trials to compare), comparing each measurement to every other measurement on the same camera. With 20 configurations and 5 trials per configuration, each camera produces a symmetric 100 × 100 matrix. The resolution/compression pairs which result in the Pareto-efficient measurements for each camera are later aggregated against the entire network.

4.4. Evaluation Results. After over 100 hours of data collection at varying times of day across two weeks, the Pareto sets for all 37 cameras are calculated (see Figure 6 for sample matrices of 8 cameras). Considering only configurations in the Pareto sets eliminates (on average) approximately half of the tested configurations as inferior and redundant.

After aggregating the resolution/compression parameters of the Pareto sets for the entire camera network, we found that, surprisingly, *every* configuration tested was in the Pareto set for at least one camera. This suggests that there is no global network-wide consensus that any camera configuration is inferior to any other; every (tested) setting was Pareto efficient for at least some camera. Calculating the percentages of the Pareto set memberships, however, reveals that the cameras tend to exhibit a "preference" for certain configurations over others (see Figure 7). This is in line with the previous observation that roughly half of the tested configurations are not preferred (less than a majority agreement between the cameras). It is not surprising to see higher percentages on configurations with either the maximum resolution or minimal compression since they already optimize at least one metric by definition. However, configurations such as 176 × 120/60% and 704 × 240/20% reveal local optimum which is potentially very useful for some practical applications of the video network. Using a more fine-tuned set of compression levels, we would likely be able to find more such points, aiding in the creation of a useful set of presets for specialized applications.

In order to evaluate the relative performance of the configurations, the measurements for each camera are normalized across all measurements on the same camera and then averaged on a per-configuration basis across all cameras using the same configuration. Figure 8 shows the relative performance of the top 8 configurations for the entire network. Intuitively, increasing either the resolution or decreasing the compression (resulting in higher bandwidth) has the effect of a reducing the frame rate, producing a more discontinuous video stream, and increasing maximum lag time. These top configurations can then be considered as candidates for a number of applications or environments. The max resolution/0 compression configuration in Figure 8(a), for instance, may be a good candidate for face recognition (so long as fast frame rate is not required), while face reconstruction may favor the max resolution/20% compression in Figure 8(c) due to its substantial increase in frame rate. An alternative approach to this general network optimization, however, is to optimize specifically for certain tasks.

4.5. Task-Based Optimization. Instead of conducting exhaustive tests to find Pareto-efficient configurations, the presented multiobjective approach can also be used to optimize network parameters for specific applications or tasks. This

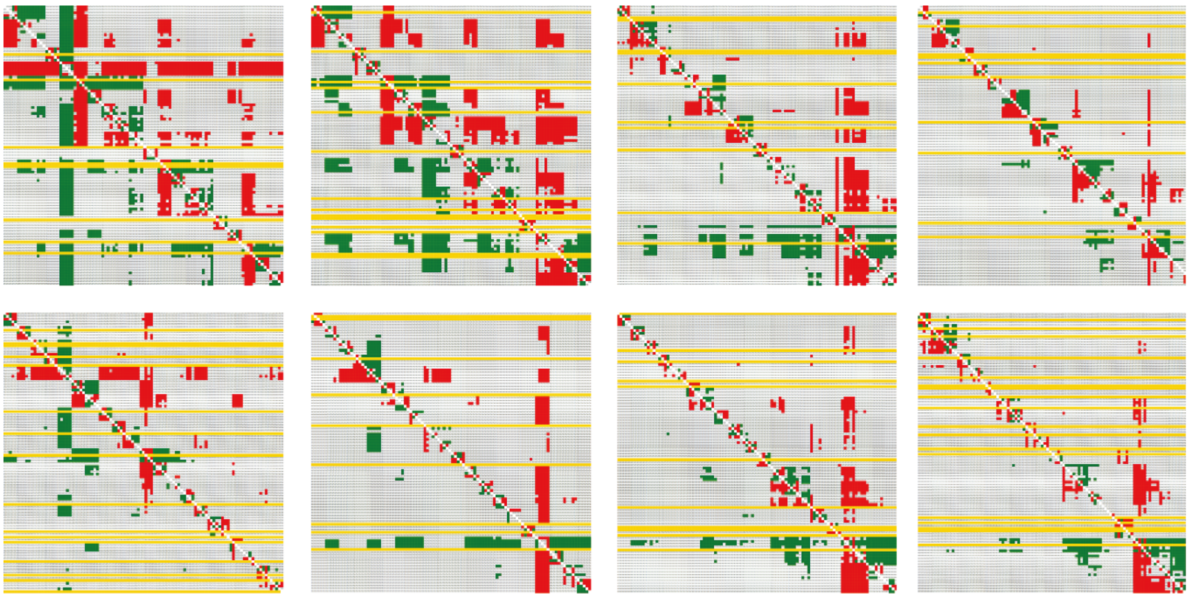


Figure 6: Measurement comparison matrices for 8 cameras. While cameras may exhibit variable performance even when using the same configurations, some configurations may be inherently better than others and exhibit similar performance across the network. To discover these configurations, 100 trials are performed on each camera under a variety of parameter configurations (i.e., resolution and compression) and each recorded measurement is compared for Pareto efficiency against the other 99 trials. This results in a symmetric matrix where vertical and horizontal axes indicate the measurements M_i and M_j, respectively (i.e., the top-leftmost square in each matrix indicates the relationship of M_1 against M_{100}). Red indicates that a particular M_i is inferior to a particular M_j, green indicates superiority, and a solid horizontal yellow line denotes rows which are completely Pareto-efficient (i.e., either superior or non-inferior against all other 99 trials).

Compression resolution\	100	60	30	20	0
176 × 120	46%	66%	46%	51%	74%
352 × 240	34%	46%	26%	34%	91%
704 × 240	51%	29%	17%	54%	97%
704 × 480	34%	31%	63%	94%	100%

Figure 7: Probability of configuration membership in any given camera's Pareto set.

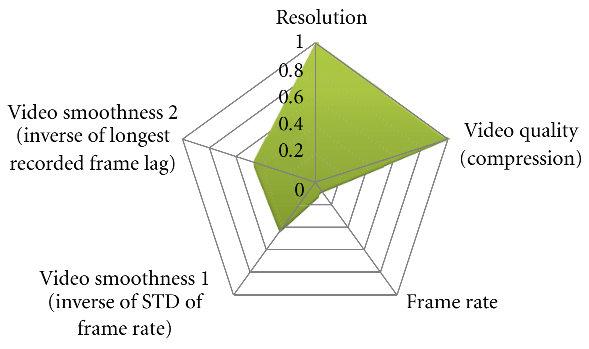
can be done in much the same way as with the other performance metrics quantifying application-specific performance (e.g., face detection rate, smoothness of tracked objects trajectories) and adding them to the multiobjective metrics. Optimizing the network for face recognition at an airport, for instance, may be done by performing the same Pareto-efficiency tests on the precision and recall rates returned by a face recognition algorithm. In order to take advantage of the video produced across all the network configurations, it is recommended to record the streams during testing so that tasks which can be performed off-line can be optimized with greater flexibility (e.g., a face recognition algorithm can be continuously tuned and repeatedly tested against the dataset without actually having to reconfigure the network). Tasks such as continuous PTZ tracking on the other hand, would have to be performed alongside the live data streaming.

5. Conclusions

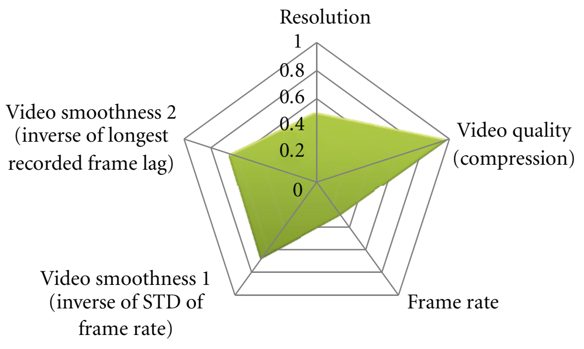
We have designed an softwarereconfigurable architecture for a wireless network of a large number of video cameras and implemented a working system by building the servers, installing the cameras, writing the software, and configuring the network to support it. Further, we gained insight into configuring the network's cameras by defining a set of metrics and discovering Pareto-efficient camera configurations by performing multiobjective optimization on a large volume of real data recorded by the system.

The idea persists that if one has a camera network with 30 FPS cameras, one will be able to obtain the said 30 frames per second regardless of network configuration or parameters. Though this may be true in a controlled test environment, the performance expectation should not be so optimistic for real-world wireless implementations. Even using the most preferred Pareto-efficient configurations on a non-congested network, it is shown that frame rates will most certainly suffer and that trade-offs must be made.

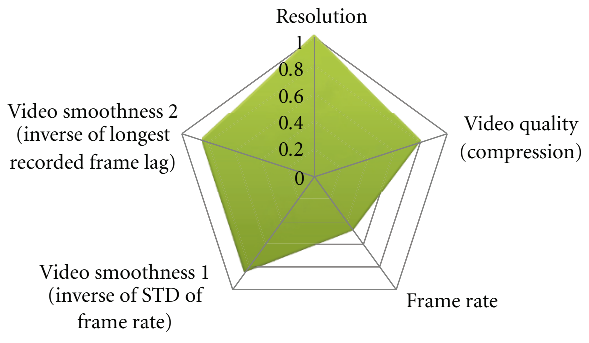
During a large workshop hosted in the building, however, it was observed that frame rates of the cameras would periodically drop and we later found that these drops coincided with breaks given during the workshop. Suspicious that a number of open and local 802.11g networks may be congesting our network, a cluster of bridges were upgraded from 802.11g to 802.11n. In daily usage, frame rates were seen to reach up to 20 FPS for even the most bandwidth-intensive configurations (such as 704 × 480 resolution with



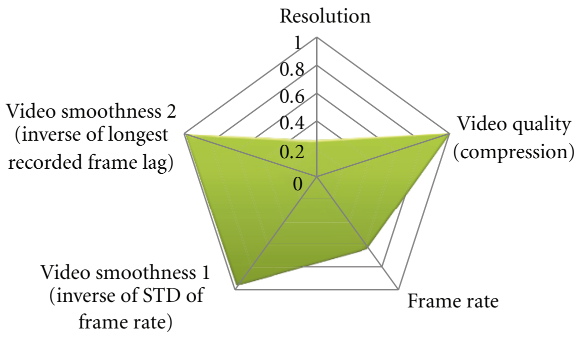
(a) 100% of cameras: 704 × 480 pixels, 0% compression



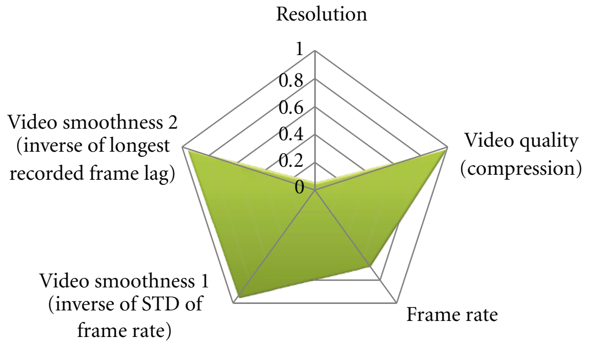
(b) 97% of cameras: 704 × 240 pixels, 0% compression



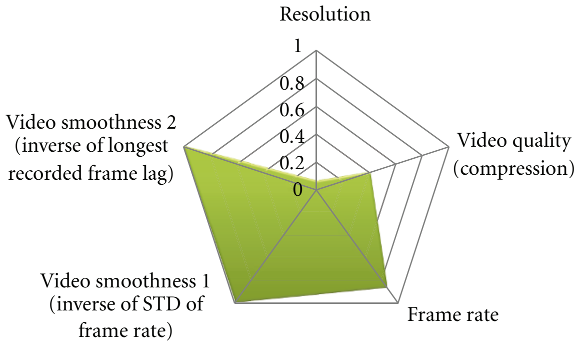
(c) 94% of cameras: 704 × 480 pixels, 20% compression



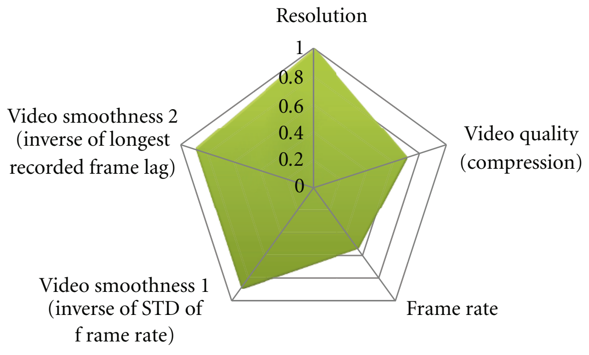
(d) 91% of cameras: 352 × 240 pixels, 0% compression



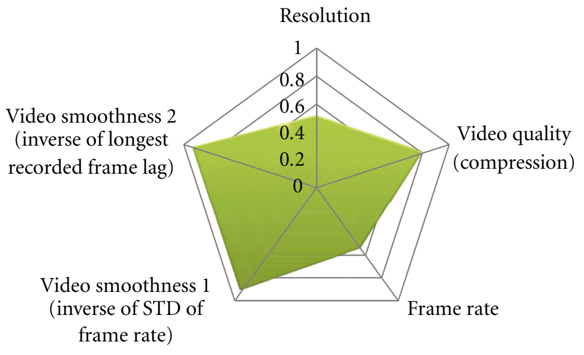
(e) 74% of cameras: 176 × 120 pixels, 0% compression



(f) 66% of cameras: 176 × 120 pixels, 60% compression



(g) 63% of cameras: 704 × 480 pixels, 30% compression



(h) 54% of cameras: 704 × 240 pixels, 20% compression

FIGURE 8: The top 8 dominating camera configurations as chosen by 37 cameras. Graphs are ordered by the percentage of cameras in which the particular configuration was Pareto-efficient and all metrics are normalized to 1.0 across all cameras. Clockwise from the top: resolution ranges from 176 × 120 to 704 × 480 (higher is better), JPEG compression settings range from 0 to 100 (lower is better, so inverse is shown), and frame rates range from 0 to 30 FPS (higher is better). For measuring the "smoothness" of outputted video, the standard deviation of the frame rate (recorded at 1-second intervals) and maximum lag time between any two sequential frames is recorded (lower is better, so inverse is shown).

0% compression) where they were previously achieving typically only 3 FPS (even when other bridges in the network were not in use). While this makes a case for upgrading to 802.11n, this also suggests that network congestion from other networks may play a large role in frame rates and that networks may wish to operate in a dedicated frequency range.

In situations when even hardware upgrades can still not achieve sufficient performance, however, we would like to emphasize that partial data is still important. Rather than having algorithms which assume that the data consists entirely of complete video frames (and are only capable of processing such frames), realtime computer vision algorithms should take advantage of as much information as is available to them; the constant stream of partial frames which may only be missing the last few rows of data can still be tremendously useful for a number of applications.

Acknowledgments

This work was supported in part by NSF grants 0551741, 0622176, 0905671, and ONR grants N00014-07-0311, N00014-07-1-0931.

References

[1] I. F. Akyildiz, T. Melodia, and K. R. Chowdhury, "A survey on wireless multimedia sensor networks," *Computer Networks*, vol. 51, no. 4, pp. 921–960, 2007.

[2] W.-T. Chen, P.-Y. Chen, W.-S. Lee, and C.-F. Huang, "Design and implementation of a real time video surveillance system with wireless sensor networks," in *Proceedings of the 67th IEEE Vehicular Technology Conference (VTC '08)*, pp. 218–222, May 2008.

[3] H. Park, J. Burke, and M. B. Srivastava, "Design and implementation of a wireless sensor network for intelligent light control," in *Proceedings of the 6th International Symposium on Information Processing in Sensor Networks (IPSN '07)*, pp. 370–379, ACM, New York, NY, USA, April 2007.

[4] P. Chen, P. Ahammad, C. Boyer et al., "Citric: a low-bandwidth wireless camera network platform," in *Proceedings of the 2nd ACM/IEEE International Conference on Distributed Smart Cameras (ICDSC '08)*, pp. 1–10, September 2008.

[5] C. Park and P. H. Chou, "eCAM: ultra compact, high data-rate wireless sensor node with a miniature camera," in *Proceedings of the 4th International Conference on Embedded Networked Sensor Systems (SenSys '06)*, pp. 359–360, ACM, New York, NY, USA, November 2006.

[6] T. Teixeira, D. Lymberopoulos, E. Culurciello, Y. Aloimonos, and A. Savvides, "A lightweight camera sensor network operating on symbolic information," in *Proceedings of the 1st Workshop on Distributed Smart Cameras*, November 2006.

[7] R. Kleihorst, A. Abbo, B. Schueler, and A. Danilin, "Camera mote with a high-performance parallel processor for real-time frame-based video processing," in *Proceedings of the IEEE Conference on Advanced Video and Signal Based Surveillance (AVSS '07)*, pp. 69–74, September 2007.

[8] M. Quinn, R. Mudumbai, T. Kuo, Z. Ni, C. D. Leo, and B. S. Manjunath, "VISNET: a distributed vision testbed," in *Proceedings of the 2nd ACM/IEEE International Conference on Distributed Smart Cameras (ICDSC '08)*, pp. 364–371, September 2008.

[9] H. T. Nguyen, B. Bhanu, A. Patel, and R. Diaz, "VideoWeb: Design of a wireless camera network for real-time monitoring of activities," in *Proceedings of the 3rd ACM/IEEE International Conference on Distributed Smart Cameras (ICDSC '09)*, August 2009.

[10] FFmpeg Team, "libavcodec: audio/video codec library," 2010, http://ffmpeg.mplayerhq.hu/.

[11] S. Fluhrer, I. Mantin, and A. Shamir, "Weaknesses in the key scheduling algorithm of rc4," in *Proceedings of the 4th Annual Workshop on Selected Areas of Cryptography*, pp. 1–24, 2001.

[12] N. Li, B. Yan, and G. Chen, "A measurement study on wireless camera networks," in *Proceedings of the 2nd ACM/IEEE International Conference on Distributed Smart Cameras (ICDSC '08)*, pp. 1–10, September 2008.

[13] S. Birchfield, "Elliptical head tracking using intensity gradients and color histograms," in *Proceedings of the 1998 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 232–237, June 1998.

[14] The MathWorks, "MATLAB," 1994–2010, http://www.mathworks.com/products/matlab/.

[15] G. Bradski, "Open computer vision library (OpenCV)," 1999–2010, http://opencv.willowgarage.com/.

[16] I. Microsoft, "Microsoft foundation classes (MFC)," 1992–2008, http://msdn2.microsoft.com/en-us/library/d06h2x6e(VS.80).aspx.

[17] C. M. Kohlhoff, "Boost.Asio: a cross-platform c++ library for network and low-level I/O programming," 2010, http://www.boost.org/doc/libs/1410/doc/html/boost asio.html.

[18] Network Working Group, "RFC 2616: hypertext transfer protocol—HTTP/1.1," 1999, http://www.ietf.org/rfcrfc2616/.txt.