Data-Centric Visual Sensor Networks for 3D Sensing

Mert Akdere, Uğur Çetintemel, Daniel Crispell, John Jannotti, Jie Mao, and Gabriel Taubin

Brown University, Providence RI 02912, USA

Abstract. Visual Sensor Networks (VSNs) represent a qualitative leap in functionality over existing sensornets. With high data rates and precise calibration requirements, VSNs present challenges not faced by today's sensornets. The power and bandwidth required to transmit video data from hundreds or thousands of cameras to a central location for processing would be enormous.

A network of smart cameras should process video data in real time, extracting features and three-dimensional geometry from the raw images of cooperating cameras. These results should be stored and processed in the network, near their origin. New content-routing techniques can allow cameras to find common features—critical for calibration, search, and tracking. We describe a novel query mechanism to mediate access to this distributed datastore, allowing high-level features to be described as compositions in space-time of simpler features.

1 Introduction

We propose an architecture for the construction and use of *Visual Sensor Networks* (VSNs). VSNs will handle much richer data than today's simpler data collection sensornets. Cameras will perform local image processing, and then cooperate to perform higher-level tasks, such as calibration, view combination, object detection, and tracking.

Today, these systems are monitored by a small army of security personnel. Smart event detection based on higher level analysis of the image data can help alleviate this burden. Combining information from multiple cameras, space-time trajectories of individuals can be computed, and suspicious behaviors can be identified. Today's multi-camera systems perform some of this integration, but do so in a centralized fashion, requiring all cameras to stream video data to a single server. These systems will not scale easily to the large networks that could provide more detailed and comprehensive coverage.

Although complete centralization in untenable, information from several cameras must be combined to make inferences about events occurring in the 3D world, such as detecting and tracking individuals or vehicles. In order to do so, the location and orientation of each camera must be determined with respect to a single coordinate system. This is the camera calibration problem. Mobile cameras require continuous calibration.

S. Nittel, A. Labrinidis, and A. Stefanidis (Eds.): GSN 2006, LNCS 4540, pp. 131–150, 2008.

To meet the needs of future applications, smart cameras must process video data in real-time, produce lower bit-rate summaries, communicate and share data with neighboring cameras, and execute collaborative algorithms in a decentralized fashion.

1.1 Example Application

Consider how a potential VSN could be deployed and used in a busy metropolitan airport. The network might include the hundreds of static cameras already in use at such an airport today, augmented with thousands of additional static cameras to gain greater coverage. Hundreds of mobile cameras attached to airport personnel and equipment may also play a role.

The VSN will provide security personnel with various ways to access the camera network. The simplest is to ask for views of any area, from any direction. Virtual views would be synthesized from overlapping views provided by the camera network's extensive coverage. Operators might choose to follow people or objects that appear suspicious, or to construct a super-resolution view of a traveler's face. Moving beyond direct human control, such a network could be programmed to draw attention to activity in a restricted area, or an activity by unrecognized personnel. Finally, we envision the network detecting high-level activities such as a traveler who has left his baggage unattended.

It is critical that operators have the tools available to assess the threats detected by the network. For example, users should be able to follow a person or object *back in time* or ask high-level questions about the past. How long has this person been in the room? Which other people has he spoken with? Based on its motion when carried, how heavy is his bag?

1.2 Requirements

In order to support applications of the type we envision, smart cameras must capture and process image data in real-time, and cooperate to make that data available to applications in a structured way.

Multi-Camera Calibration and Time Synchronization. Smart cameras must share a common global coordinate system in order to combine information from disparate cameras in 3D. *Multi-camera geometric calibration* is an active research topic—current solutions are complex, involving cumbersome procedures to overcome the unavoidable partial occlusions, and are based on centralized computation, usually requiring factorization of very large matrices [1,2]. The most common approach is based on *structure from motion* algorithms, in which the pose of all cameras and the location of feature points in 3D are simultaneously estimated. VSNs, on the other hand, require a new robust solution based on distributed algorithms. Furthermore, VSNs with dynamic nodes require new, incremental approaches. *Photometric calibration* is also necessary to account for inevitable differences in sensitivity to light and color between any two image

sensors. Finally, VSNs must compensate for the lack of precise *time synchronization* at the frame level.

Virtual Views. Virtual views are images created by integrating visual data from several cameras to simulate the view of a virtual camera where none exists. These views are generated by interpolating sample values from the *light field* [3,4], an abstraction that represents the mapping of 3D rays to colors. In a VSN the individual images that constitute the light field samples are best stored in a distributed fashion near the smart cameras where they are observed. An *image-based routing* protocol efficiently routes virtual view requests to the appropriate smart cameras and composes their individual contributions within the network to minimize network traffic.

Virtual views may also be specified in *resolution* or *time*, leading to virtual video streams. By combining results from several overlapping cameras, a virtual camera of greater frame-rate or resolution may be simulated. Generating video streams consisting of frames defined by similar parameters can be less taxing since the image-based routing mechanism may cache recent decisions. Of particular interest is a virtual video stream that follows a chosen target.

Detection and Tracking. In order to track moving objects, the objects must be segmented from their background. This operation requires a continually maintained model of the background. Once foreground objects are segmented out of the background, noise removal and connectivity analysis defines *blobs*. Tracking 2D blobs over time requires a significant amount of computation at the smart camera level, but reporting their trajectories requires very little communication. Tracking in 3D requires establishing correspondences between blobs detected in separate smart cameras, requiring fine-grained calibration and collaborative processing.

Establishing correspondences between large blobs detected in different images usually reduces to feature detection and matching [5,6]. Features are small blobs which are likely to have a similar appearance in a different image. Features might correspond to corners of buildings, or facial features of people. In general, the feature data interchanged between cameras is not large, but the complexity of feature matching is in principle quadratic in the number of cameras.

Compression. Transmitting high resolution images and high frame-rate video requires a significant amount of bandwidth, and as a result, consumes a significant amount of power. However, multiple cameras capturing multiple views of the same scene produce images and video with significant redundancies. Transmitting this redundant information is a wasteful allocation of the most precious resource of battery operated wireless sensor networks. One approach to remove the redundant information is to reconstruct the 3D structure of the world and transmit a 3D video stream, which can later rendered from an arbitrary point of view. Also, surveillance applications do not require high resolution everywhere in the field of view of the cameras, but just around the detected objects. Image and video compression schemes that interact with object detection algorithms have the potential to reduce the bandwidth utilization significantly.

1.3 Challenges and Contributions

The requirements of a VSN go beyond the techniques developed for existing sensornets for two reasons. First, the raw data is extremely bandwidth intensive. Few sensor systems tackle this challenge. Those that do focus on data types that can be compressed in isolation by, for example, Fourier transform. Second, the image data is difficult to aggregate. Existing systems build collection trees in which aggregation reduces the size of acquired data at join points.

In order to aggregate image data, extensive communication must take place first. Nearby cameras must share image features in order to establish correspondences that create a common coordinate system. Even with aggregation, we expect that in-network storage will be critical to reducing bandwidth requirements. With in-network storage comes challenges in routing and distributed query processing. Our contributions lie in a scheme for storage and processing of data in the VSN, and a high-level data access mechanism for operating on that data.

3D Data-Centric Storage, Routing, and Processing. We introduce data-directed localization to dynamically calibrate without specialized hardware. Nodes will dynamically build ever larger Geographic Hash Tables (GHTs) in which the nodes share a common reference frame. GHTs allow for distributed feature matching and feature matching in smaller GHTs is used to bootstrap localization.

We also introduce data-centric processing (DCP), which places processing elements in the network, located where the data they process will be stored in the GHT. These processing elements operate on data as it becomes available, inserting new, higher-level items into the datastore. Further processing elements may continue this process to produce ever more complex observations.

VSNs must support queries that seek image data for a given object from a given direction. To support these queries that do not map easily to a hash-based content routing scheme, we have developed Image Based Routing. IBR uses a *binmesh* to succinctly represents the views of many cameras in a single summary. Query routing follows the binmeshes down the routing tree toward cameras that observe the target object.

Space-Time Database Abstraction. Our work is intended to simplify the development of 3D sensornet applications in two ways. First, we use a *a space-time "cube" abstraction* for declarative access to the data available in the sensornet. This abstraction hides the raw data acquired by the cameras, providing a form of physical data independence. Applications will "pull" data using SQL-style declarative queries posed on top of the cube abstraction. Second, we rely on a predicate language for specifying space-time *feature patterns* for search and tracking of complex objects and events easily.

Three-Dimensional Hardware. We advocate stereo smart cameras—devices that will include two imaging sensors. The second image sensor adds little to the cost or complexity of the design, but enables significant 3D sensing functionality as well as a reduction in bandwidth utilization by leveraging the 3D structure of the data.

2 System Model

In comparison to existing sensornet approaches, our approach concentrates heavily on in-network processing and storage. Existing systems generally use two techniques: push computation all the way to the sensors (*i.e.*, avoid repeating redundant observations), or thin data by aggregation along a collection tree (*i.e.*, by averaging reading). Our techniques are more cooperative, and less hierarchal. The nodes of VSNs must share the data collected in early stages to enable later staged. Calibration information must be shared to enable feature detection and tracking. Features (and their importance) must be shared in order to guide compression. Collection along a tree is insufficient. Our basic system model is illustrated in Figure 2.

1. "Interesting" data is identified using complex query and event specifications over a space-time database (Sect. 5).
2. Queries and event specifications on application-level features (*e.g.*, a moving person) are translated into an execution plan. The constituent feature-oriented query operators are deployed to allow data centric processing.
3. Image features are extracted locally at each camera. They are then propagated in a neighborhood, using a GHT-based model, to perform localization and calibration in a distributed, collaborative fashion. Furthermore, features are composed to form higher-level features using the data centric execution plan.
4. Raw camera data is acquired, compressed, and stored locally (not necessarily on the source camera). The compression is enhanced because of calibration and feature detection as image redundancies and unimportant data are eliminated. Most sensor networks seek only to produce output for external consumption. VSN nodes must share their computation (here, calibration data and features) in order to function more efficiently (here, compress better).
5. Camera data is retrieved either as direct output from queries, or in a raw form, but identified by the queries (e.g., "here is the trajectory of the car you asked about" or "here is the face of the person who left his bag in the atrium"). To meet potential resource constraints, raw data is compressed

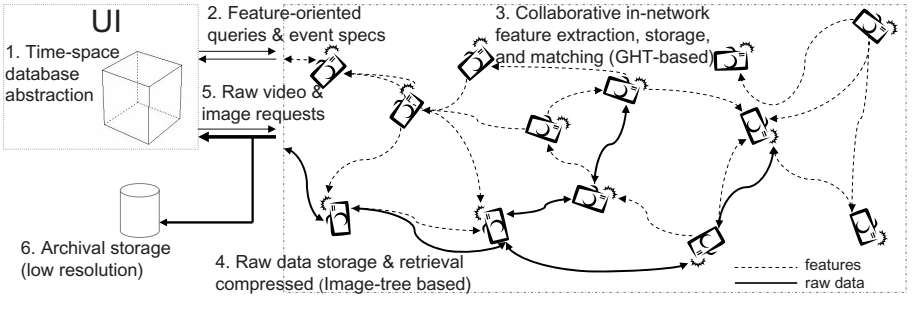

Fig. 1. High level Visual Sensor Network Model

such that interesting features are kept at a higher resolution at the expense of other features.
6. For applications that require it, all data is extracted in compressed, low-resolution format and archived for forensic/historical analysis and legal compliance. Network capacity will limit the fidelity of this data, but calibration and feature knowledge will greatly improve upon the fidelity that might be obtained by individual video streams from each camera.

3 Related Work

Visual Sensor Networks operate at the intersection of many fields, including image processing, traditional wireless networking, and distributed data management.

3.1 Centralized Image Processing

Most existing multi-camera systems are centralized, with all cameras streaming video to a central server where the data is analyzed and visualized [7]. Most early systems focused on the data management benefits resulting from the transition from analog to digital and use the networking infrastructure only as a transport layer [7]. They do no collaborative processing [1,2]. These approaches do not scale to systems with large numbers of video sensors. To address scalability, most of the data intensive processing must be performed at the source, with low-bitrate, highly compressed descriptions of the data transmitted off node. As described above, collaborative processing may result in higher levels of data compression and additional savings in power and bandwidth utilization.

3.2 Routing in Sensornets

A wide variety of protocols have been developed for ad-hoc wireless routing [8,9,10]. These protocols attempt to produce a traditional network-layer, allowing hop-by-hop communication between named end-points. As such, they are not appropriate for the needs of sensornets in which node identity is rarely important.

Instead, specialized protocols have been developed with sensornets in mind. Trickle [11] supports the dissemination of code to a sensornet while minimizing communication costs. Data Centric Storage, in the form of a Geographic Hash Tables [12] (GHTs), has been proposed to allow the storage and retrieval of named data items within the network. A GHT stores a data item by hashing its name to a geographic coordinate and then storing the item (or a pointer) at the node closest to that coordinate. A modified version of GPSR [13] is used to route the item to the nearby node and several replicas. In this work, we extend GHTs in a number of ways adding new support for features that are important to processing data in-network.

3.3 Abstractions for Wireless Sensornets

High-level interfaces for application development in sensor networks have received significant recent attention. Most work in this direction focused on basic operating system and communication support [14], neighborhood and abstract regions [15,16], data-centric event dissemination [17,18], and multi-resolution data storage [19].

Closest to our work are those that take a database-centric approach to sensor network data access, such as TinyDB [20] and Cougar [21]. Recent work [22] has addressed space-time queries in sensornets. These systems focus on aggregation style queries over scalar numerical values, whereas our proposal focuses on a richer space-time database over multi-dimensional image/video data, requiring major changes in the way queries are expressed and executed in the network. In addition, these systems were designed to deal with low-rate data whereas VSNs must handle significantly higher data rates, which we address with novel in-network computation and 3D compression.

3.4 Visual Sensor Networks

The importance of Visual Sensor Networks has been recognized in the broad Sensor Networks literature [23,24], but only relatively small testbed systems, most often wired, have been constructed [25,26]. Similar systems have been proposed for surveillance and security applications, urban traffic control [27], and many more for military applications (refer to [28] for various references). A number of systems for image-based-rendering applications have been proposed including a moderate number of cameras arranged in a regular fashion [29], which can produce super-resolution in time [30]. All of these algorithms are centralized. They require the frames of all cameras to reside in a single place. Attention has not been paid to distributed algorithms applicable to the VSN framework. The same is true for stereo and multi-camera calibration algorithms [1]. Establishing feature correspondences can be done in a pairwise fashion, but it is prone to errors. Robust algorithms such as RANSAC [31] have proven reliable.

4 Network Protocols and Coordination

Visual Sensor Networks have several unusual requirement as compared to traditional wireless networks, or even existing sensornets. Camera networks require fine-grained calibration, distributed feature matching and search, and image oriented routing techniques.

4.1 Data-Directed Localization and Synchronization

We are not the first to observe that sensor networks are data-oriented [32]. In existing sensor networks, requests are routed to the sensors best able to make a specific observation by geographic routing techniques. For example, to find

the average temperature in a room, a request is routed toward the room, and then to all nearby sensors. In a static network, this routing might be based on the known locations of immobile sensors. In a mobile network, dynamic routing protocols determine the set of nodes in the target area, usually with the aid of localization hardware such as Cricket [33] or GPS.

In visual sensor networks, the need for localization must be generalized to include orientation and field of view. Small angular errors in orientation may be unacceptable when a distant object's location is estimated, or the views of two cameras are to be integrated.

We propose *data-directed localization* in which smart cameras will localize with respect to each other based on observed image data. Existing localization techniques are not accurate enough to allow for image aggregation. Even the best localizers have only centimeter scale accuracy which is insufficient for accurately determining the orientation of a small sensor node.

In data-directed localization, sensors nodes will detect local features and then cooperate to find common features observed by multiple cameras. When nodes share multiple features, they will orient themselves in a shared coordinate system. Additional cameras will orient themselves in this system by finding features in the shared space. Time synchronization may be accomplished in the same way by the shared observation of temporal events.

Local Feature Detection. Feature detection begins as a local level 2D operation. Next, 3D features can be calculated locally from 2D features by using two image sensors separated by a known baseline. The inter-camera search for correspondences is drastically reduced by using 3D features. Sublinear geometric matching techniques exists for spatial configurations of small groups of features in 3D [34]. These advantages motivate the use of 3D features to reduce bandwidth and power utilization and explain why we advocate the use of smart cameras with two image sensors.

Distributed Feature Matching. Distributed feature matching can be built on top of the idea of a Geographic Hash Table [12]. After detecting local features, each camera using geometric (*not* geographic) hashing [34], to bin them into similar categories. Once categorized, these features are inserted into the GHT, keyed by category. Similar features will therefore be placed at the same location. Prospective matches can be determined at that location, and the nodes with shared features can be notified, allowing them to directly confirm the match and calculate their relative transformation matrix.

Unfortunately, GHTs rely on a preexisting localization scheme to enable geographic forwarding. In order to forward objects to their hashed locations, the nodes must know their own locations and the locations of their neighbors. But we intend to use feature matching in order to determine node locations!

Bootstrapping GHTs. We address this difficulty by bootstrapping localization in small GHTs and extending GHTs to allow merges. Nodes will first organize in small GHTs in which features may be advertised by scoped flooding over n hops. Any two nodes that are within n hops of one another and share features will

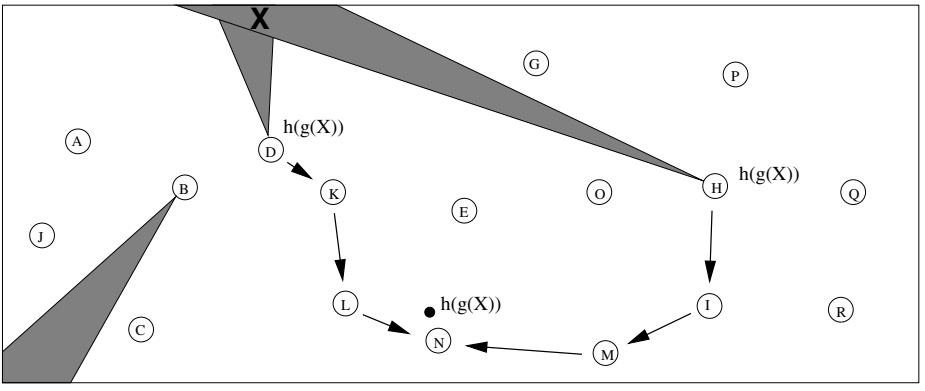

Fig. 2. The feature X is observed by the two distant camera nodes. The feature is categorized through a geometric hash function, g(), and then a storage location is selected with a geographic hash, h(). Each camera routes the feature toward the designated location, where the closest node, N, stores the feature, detects matches, and informs the observers.

detect their overlap and orient themselves in a mutually agreed upon coordinate system. When nodes in separate GHTs detect overlap, the GHTs merge using a single coordinate system. Previous sensornets are incapable of bootstrapping localization in this way because they do not sense distant features in sufficient detail to determine precise relative positions.

In order to merge adjacent GHTs into a single unified coordinate space, shared features must be discovered. However, these features may be shared among nodes that are too far from each other to find each other with scoped broadcast. However, now that small GHTs have been established, they can be used to find more distant matches. A GHT member may be in radio contact with nodes in another GHT. Border nodes from one GHT may place features into the adjacent GHT with knowing the relative transform between the GHTs. Since the GHT will now contain its own features and the features of the adjacent GHT, matches can occur between members of both GHTs.

Feature matching is a useful primitive for tasks beyond localization. To implement tracking, adjacent cameras must realize they are observing the same object. To generate virtual views, multiple views of the same object must be aggregated. Further, we expect most searches in a three dimensional object space to be example based. Such searches can be viewed as feature matching between an abstract target and features detected in the environment. We turn to the task of these general searches in the next section, using feature matching as an important primitive.

We have developed a novel auto-calibration algorithm to estimate the relative position and orientation of several *camera pods*, each consisting of four rigidly mounted network cameras. A processing engine simulated in a computer cluster converted each pod into a multi-sensor smart camera. In addition, basic 3D tracking was demonstrated using the estimated camera parameters. Figure 3 shows

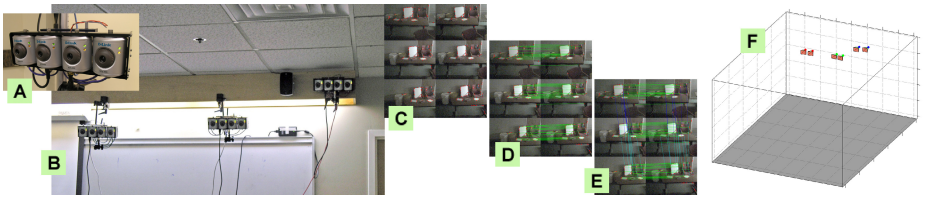

Fig. 3. Preliminary results using proposed 3D feature matching approach: (A) Camera pod; only two of the four cameras used in the calibration algorithm; (B) Experimental setup with three camera pods; (C) features extracted in each image independently; (D) 2D short baseline feature matching within each camera pod results in 3D features; (E) 3D feature matching between camera pods reduces matching complexity; (F) calibration results.

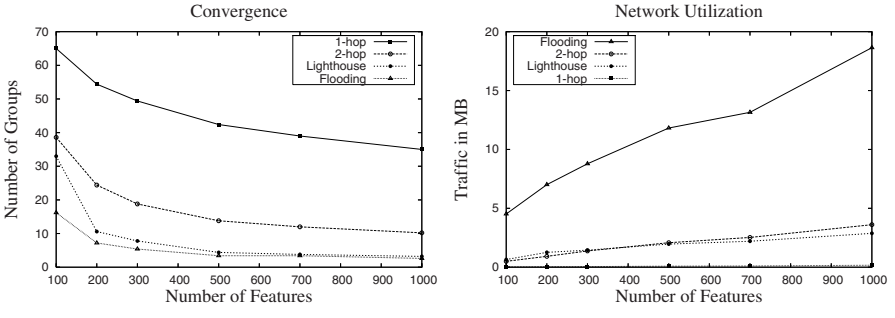

Fig. 4. Simulation results for distributed matching using incrementally built GHTs shows that the GHT scheme exhibits convergence performance comparable to complete feature flooding, but uses little more bandwidth than a simple 2-hop feature propagation protocol. In both graphs, low numbers are better. On the left, they reflect convergence into fewer individual coordinate systems. On the right, the reflect lower network utilization

the pods mounted in the laboratory, and illustrates the steps of the algorithms. Each pod performed small baseline feature matching and stereo reconstruction to construct 3D features. Next, these 3D features were matched between pods. Using a minimum of three correspondences, the pods calculate the rotation and translation necessary to bring themselves into a common reference frame.

In parallel to this exploration of a centralized stereo matching technique, we have simulated the performance of a distributed matching protocol, Lighthouse [35]. Lighthouse attempts to converge an uncalibrated camera network into a single coordinate space using the GHT matching techniques described above. These simulations show that Lighthouse finds distributed matches nearly as well as complete feature exchanges, yet uses less bandwidth than exchanges among 2-hop neighbors. In these simulations of 100 camera networks, complete feature exchange uses approximately five times the bandwidth of Lighthouse. This advantage grows as the network increases in size.

4.2 Feature-Oriented Search and Computation

Visual sensor networks will gather vast amounts of data that must be searched, processed, and acquired by users and applications. GHTs were proposed in traditional sensornets as a compromise between moving all acquired data to a central site, and leaving data at its point of acquisition. Centralizing data requires enormous network capacity and power for transmission, regardless of whether the data is ever queried. Leaving data unindexed at the acquisition site is costly because queries must conduct exhaustive search to locate a data item.

The hash function of a GHT can be thought of as an index on arbitrary data. If data is stored using names that correspond to the needs of queries, retrieval and processing are efficient. For example, if cameras detect and measure the heights of people they observe, they might store these observations (or pointers to them) keyed by those heights, binned in one inch increments. A query can find all individuals greater than six feet tall by examining the hash locations associated with each potential observation above six feet. Sect. 5 presents a relational database abstraction to sensor data, and just as in an RDBMS, we will support arbitrary indexes by storing data according to hash functions that correspond to expected queries.

Data indexed in this way is first hashed into a category, or bin. Next the GHT applies a hash function to select a location for the category. Significant query performance may be lost due to the random locations selected, even for keys that will be queried sequentially. For example, suppose that an application seeks observations of faces in a room—a specific geographic area. These observations might have been inserted into the GHT with keys like, $\langle \texttt{face}, x, y \rangle$, where x and y are the geographic coordinates of the observation, rounded to categorize the observations. The query must lookup each possible value for x and y for coordinates in the room. Hashing each such key results in the storage of these observations arbitrarily throughout the sensornet.

Locality Preserving Hints. We propose widening the interface to the GHT's insertion operation to include an optional coordinate "hint." The GHT hash function will use the supplied coordinate to directly set the high bits of the coordinate at which the data will be stored. In the common case of observations with spatial locality that will be accessed by location, geographic hints will preserve spatial locality and allow queries that access the observations sequentially to operate at a small set of nearby locations. In the example above, observations with similar x and y coordinates will be stored near each other.

Generalizing, we will also allow hints containing a small number of arbitrary scalar values. By taking these values into account during hashing, a set of linear values can be hashed along a line in geographic space. Multiple values can be hashed to a two dimensional patch. Suppose facial observations of faces are stored with hints describing the distance between eyes and the width of mouth. The hinted hash of these observations will map them to a single quadrilateral in the sensornet. Queries that must process a range of values will exhibit spacial locality as they retrieve and process observations in the network.

Feature Aggregation with Data-Centric Processing. The detection of high-level features is generally accomplished by detecting simpler features in a particular arrangement. Primitive feature detectors place a record of their finding in the GHT by inserting the feature under a well-defined name, such as `eye`. To detect higher level features, a second level of feature detectors can be located on the nodes that will receive the individual subfeatures. For example, at hash(`mouth`), a face detector notes the location of the mouth and inserts a partial face observation in the GHT. A similar aggregator creates partial observations for eyes and noses. These observations are all inserted under the well-known name `partial-face` at the same location. When enough observations agree, a face has been detected, and the composite event is inserted at hash(`face`). We consider these operators, placed in the GHT to process values at their insertion point, to be the natural computational analogue to data-centric routing and storage—data-centric processing.

Scoped GHTs. The GHT abstraction provides precise insertion and lookup operations. That is, if any node inserts data under a given key, a lookup from any other node will find the given item. However, feature matching and aggregation do not require this strong guarantee. There is no need for `partial-face` observations associated with eyes observed hundreds of meters apart to be stored at a single point. We propose Scoped GHTs that perform insertions nearby in the case that the observation need not be accessible from afar.

We will explore a geographic hashing scheme, inspired by our previous work on GLS [10], that enable this relaxation. We will divide the world with a fixed sized grid, and store features only in those squares where accessibility is needed. If the feature is used only in queries that require it to be observed within two feet of another feature, then the feature is inserted only in those squares that are within two feet of the feature's location. Scoped insertions require constant power and bandwidth, dependent on square size, rather than the $O(\sqrt{(}n))$ power and bandwidth required to traverse a sensor field of n nodes to a random location.

Figure 5 shows how DCP, using scoped insertions is improved. On the left, observation are sent to a central base station in order to perform complex event detection. On the right, features are are sent only to local base stations. On average, features are tranmitted shorter distances, and the load is spread among many stations. Nonetheless, all complex features are still found, since subfeatures that may be a part of the same larger feature are collected at the same base station.

4.3 Structured Routing and Aggregation

Until now, we have discussed techniques that allow the VSN to perform tasks, such as calibration, feature detection, and tracking, without transmitting large amounts of visual data. Howevere, the VSN must also return visual data efficiently when requested.

Image Based Routing. A sensor network must support data directed queries such as, "Show me the view of the Atrium, from the North." The querier does

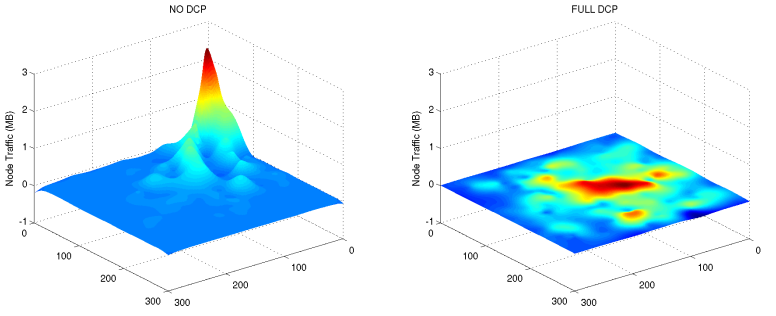

Fig. 5. Network transmissions are plotted as "heat map." On the left, transmission to a centralized base station overwhelm the nearby nodes. On the right, DCP with scoped insertions balances the load to provent hotspots.

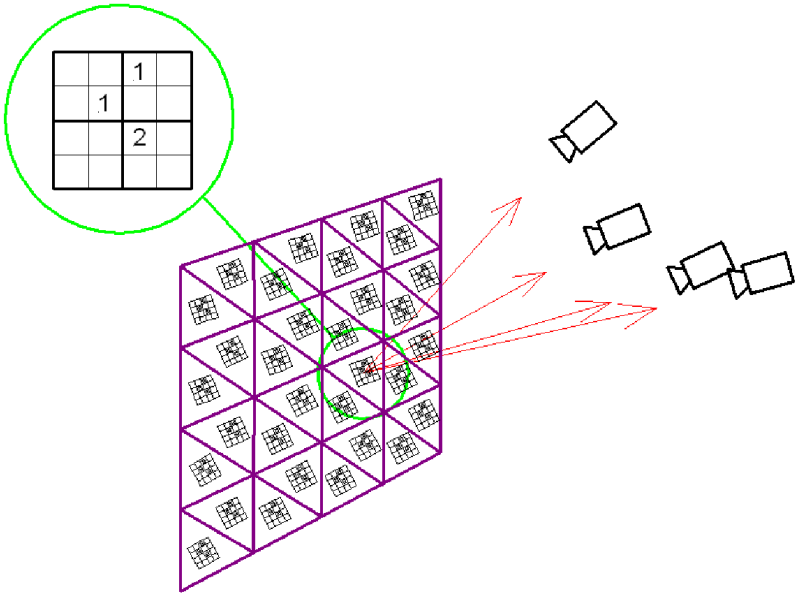

Fig. 6. To build binmeshes, the observable geometry is divided into tiles. A faceted hemisphere is placed on each tile. A *binmesh* describes the set of facets observable by a camera or set of cameras.

not know or care which sensors are involved in answering the query. In a visual sensor network, queries must be routed to the sensors that can observe an area, rather than the sensors in the specific area.

We are integrating our work on Image Based Routing into our VSN framework. The IBR protocol is used when a query is seeking data for a specific location as seen from a specified direction. Image Based Routing resembles a traditional distance-vector routing protocol with route aggregation, rather than

hash-based content routing. Image Based Routing is tree based. The leaves of the tree pass descriptions of their fields of view to their parent. As the descriptions work their way toward the root, nodes aggregate multiple view descriptions into a single description that describes their own view and the views of their descendants. A query for any particular view can be routed from the root by choosing the child(ren) that has advertised a view description matching the query. Responses to the query are aggregated from partial responses as they flow back toward the root from the various responding cameras.

View Representation. We have developed the *binmesh*, a datastructure which represents the angles from which a given camera observes a given geometry. Aggregating these binmeshes is, approximately, a bitwise OR. Once aggregated, accuracy remains high. An aggregated binmesh does not represent any new impossible views that its constituent binmeshes did not declare. **Figure 6 is neat.**

5 Data Access and Querying

An important goal of our proposal is to simplify VSN application development. To this end, we will allow users and applications to ask questions about the network in a high-level language that specifies what data to gather from the network without specifying how the query should be executed. The system must adpatively plan in order to execute the query efficiently, taking into account other simultaneous queries, and the fidelity needs that may influence compression.

5.1 Space-Time Database

Our primary abstraction is a space-time 4-dimensional view of the underlying data, consisting of a 3-D volume (x,y,z) representing geographic space and the fourth dimension t representing time. Conceptually, this abstraction captures the data produced by all sensors in a sequence of frozen time frames, where each frame is a 3-dimensional cube that provides a logical model of the world of interest. This abstraction allows users to easily query the system based on spatial attributes on a combination of live and historical data. This view is virtual and not materialized. The implication is that whenever a query is asked on this view, the execution involves accessing the base data stored in a distributed manner in the network. Similar virtual view (but *non* space-time) abstractions have been used by Cougar [21] and TinyDB [20].

Multi-Level Data Representation. Our framework is based on a two-level representation of sensor data: The *raw data layer* and the *view layer*. The raw data layer is the physical layer that continuously acquires and stores camera data. The view layer is the logical layer that transforms raw sensor data into the cube abstraction. User queries are executed on this abstraction. This layering provides physical data independence, a key concept borrowed from traditional relational database systems, which shields users and applications from the details of our data-centric protocols.

User queries can be one time or continuous and can be saved as named views that can be reused once defined. This style of cascading is similar to the way that views are cascaded in traditional database systems. In simplified terms, the semantics of cascading of a query q and a view v is that the output resulting from the execution of v will be fed into q. Cascading simplifies the expression of complicated queries and allows the same underlying query to be used concurrently by multiple others, facilitating resource and result sharing. Furthermore, multiple cascading queries allows for interesting cross-query optimizations (*e.g.*, pushing decimation operations present in the high-level query to the underlying query during execution).

Data Access Methods. The basic data access and querying interface will be a linear, SQL-like notation from declarative queries. Consider the following example query:

```
SELECT from CUBE
 WHERE location = [(50,50,50), (100, 100, 100)]
 VIEWPOINT = (100, 100, 100)
 WITH RESOLUTION 20 fps
 SAVE AS VIEW ‘‘Atrium NE’’
```

This continuous query selects a volume of space specified by two corners of a sub-cube and asks for an image stream that corresponds to the target volume as observed from a specific viewpoint. The data are to be acquired with a temporal resolution of 20 frames per second. If the viewpoint cannot be presented due to lack of data, the system might offer an alternative but similar viewpoint for which data is available. The query is saved as view “Atrium NE” as it provides a view of the room named Atrium from the North-East direction. On top of this view, we can define another query that returns images containing bag-like shapes with a resolution of ten frame per second:

```
 SELECT  from ‘‘ATRIUM NE’’
 WHERE object = ‘‘bag’’
 RESOLUTION 10 fps
```

We envision moving beyong a textual language to a graphical tool to allow incremental visualization of the sensornet data. The interface of the tool will resemble the familiar mapping software in that it will allow users to graphically select geographical spaces, zoom in and out, pan in different directions, as well as provide more advanced features such as selecting arbitrary viewpoints, and looking back in time. The operations specified through the visualizer will be translated into queries and submitted for execution to query the sensornet.

Space-Time Feature Predicates. Any interesting VSN will require search and detection of objects and activities (events) based on images. A user might be interested in finding where a specific person was at a specific point in time, locating all bags of a certain color, size, and shape, or even ask to see the people that

are currently running. In general, there is a clear need for an extensible programming framework that will facilitate the specification of objects and activities of interest.

Our framework will give the users the ability to specify spatial and temporal features on image data. We uniformly represent both objects and activities using features. Spatial features are defined based on relationships of data over space. For example, a head can be described using a spatial relationship among other (lower-level) features such as eye, nose, ear, mouth, etc. Such relationships are models that represent a feature using the relative spatial orientation of one or more lower-level features (*e.g.*, a nose is below the eyes and above the mouth). Users will register predicates that evaluate both primitive features (*e.g.*, a nose or an eye) and composite features (*e.g.*, head, body, person). Temporal features will be defined in a similar manner although, in this case, one is interested in the position of features over time. For example, the activity of "moving" can be expressed as a specific feature changing its location over time.

Clearly spatial and temporal predicates can be intermixed to express arbitrarily complex objects and activities: a running person can be identified by evaluating the spatial predicate that detects a person in a given time snapshot and then a temporal predicate that checks whether that person is moving faster than a given threshold. Once defined, the predicates will be sent to the network locations where they will be evaluated with data centric processing. Defining objects and events using feature predicates will allow us to use the feature-based routing and matching techniques outlined in Sec. 4.1 as the uniform underlying in-network query execution mechanism.

We have built a space-time database layer that facilitates queries over the location and trajectory of moving objects. Object positions acquired using cameras and Cricket nodes are stored in a centralized database and organized in multiple orders to facilitate efficient space-time queries. This work is explores sophisticated interactions with people and various artifacts in a large museum setting. This Smart Museum project will continue to serve as a target application for our work.

Adaptive Query Execution. Once the user submits a query to the system, the query will be translated into an execution plan. The planning phase must decide, based on the query specification, which routing indexes (spatial, temporal, feature-based, or a combination) and feature detectors to use. The query plan will be sent to the network and executed collectively by the appropriate nodes (as described earlier in this section) in a distributed fashion, after which results are sent back to the user. This is a distributed query optimization problem and is one of the main challenges that we will tackle.

Query execution must also *adapt* to dynamically changing workload and network characteristics. Our primary tool for adaptation will be *application-aware compression* where the novel compression techniques discussed earlier will be applied to camera data selectively based on their utility to the existing queries and the availability (or lack of) resources. The issues we will address include how to extract utility information from the existing workload and how to use this information to guide compression decisions.

5.2 Image and Video Stream Compression

We regard 3D reconstruction as a mechanism to compress the data captured from multiple video streams, resulting in lower power consumption. Polygon Meshes and oriented point clouds are popular representations for 3D surfaces in computer graphics. Polygon meshes are highly compressible [36,37,38,39,40,41,42,43] , and point clouds are error resilient. In the context of VSNs a representation with the two properties is needed. We propose a new surface representation composed of time-dependent overlapping parameterized surface patches, or *3D video streams*.

3D video streams as compression of multiple 2D video streams. Traditional video standards such as MPEG-4 support the transmission of a dynamic 3D scene as a set of multiplexed video streams, one per camera. As the number of video streams grows under constant channel capacity, the overall image quality decreases. We advocate the generation of a compressed representation of all possible views, exploiting the correlation between multiple views of the same scene. Desired views are rendered at the terminal. More computation may be required both at the encoder and decoder, but overall distortion will be minimized, and power consumption due to data transfer will be significantly reduced.

Adaptive 3D stream and Video Sampling. Figure 7 illustrates the way our compression may leverage feature knowledge obtained by the VSN. This will extend the work of Balmelli, Taubin, and Bernardini [44] from static textured polygon meshes to 3D video streams. A smart camera captures a high resolution video stream. In cooperation with other cameras, queries detect and track objects, such as faces or suitcases, resulting in a monochromatic *alpha channel* which assigns *importance values* to different pixels. The smooth alpha channel can be downsampled quite aggressively. Each of these decimated frames is used to generate a 2D *warping function* used to resample the frames of the source video stream adaptively on a pixel grid of the same dimensions as the downsampled alpha channel. The result is transmitted as an RGBA video stream of low resolution and normal frame rate, which can be further compressed using standard methods. These low resolution images preserve the details of the regions of interest at the captured resolution. In the decoder, the inverse warping function is computed from the alpha channel, and the unwarped image is recovered at the full

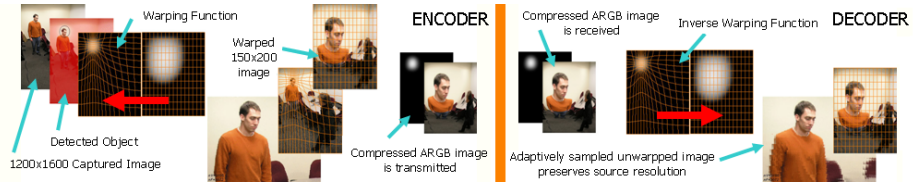

Fig. 7. Video Compression for Surveillance will use adaptive sampling and downsampling

resolution in the important regions. Again, by sharing the results of prior stages (feature detection) greater efficiencies can be obtained later (in compression).

6 Conclusions

VSNs represent an opportunity and challenges. Smart cameras offer far richer capabilities than simpler sensors, but require far greater effort to coordinate effectively. We have outlined a vision for using camera networks effectively, from the initial problem of self-calibration, through feature and image retrieval, to an expressive and efficient query language for application interaction.

References

1. Svoboda, T., Martinec, D., Pajdla, T.: A convenient multi-camera self-calibration for virtual environments. PRESENCE: Teleoperators and Virtual Environments 14(4) (August 2005)
2. Wang, F.Y.: An Efficient Coordinate Frame Calibration Method for 3-D Measurement by Multiple Camera Systems. IEEE Transactions on Systems, Man, and Cybernetics – Part C: Applications and Reviews 35(4), 453–464 (2005)
3. McMillan, L., Bishop, G.: Plenoptic Modeling: An Image Based Rendering System. In: Siggraph 1995. Conference Proceedings, pp. 39–46 (1995)
4. Levoy, M.: Light Field Rendering. In: Siggraph 1996. Conference Proceedings, pp. 31–42 (1996)
5. Mikolajczyk, K., Schmid, C.: A performance evaluation of local descriptors. IEEE Transactions on Pattern Analysis and Machine Intelligence 27(10), 1615–1630 (2005)
6. Lowe, D.: Distinctive image features from scale invariant features. International Journal of Computer Vision 60(2), 91–110 (2004)
7. Valera, M., Velastin, S.: Intelligent distributed surveillance systems: a review. IEE Proceedings on Vision, Image, and Signal Processing 152(2), 192–204 (2005)
8. Perkins, C., Royer, E., Das, S.R.: Ad hoc On demand Distance Vector (AODV) routing. Internet draft (work in progress), Internet Engineering Task Force (October 1999)
9. Johnson, D.B.: Routing in ad hoc networks of mobile hosts. In: Proc. of the IEEE Workshop on Mobile Computing Systems and Applications, pp. 158–163 (December 1994)
10. Li, J., Jannotti, J., Couto, D.S.J.D., Karger, D.R., Morris, R.: A scalable location service for geographic ad hoc routing. In: Proc. ACM/IEEE MobiCom (August 2000)
11. Levis, P., Patel, N., Culler, D., Shenker, S.: Trickle: A self-regulating algorithm for code propagation and maintenance in wireless sensor networks. In: NSDI 2004 (March 2004)
12. Ratnasamy, S., Karp, B., Yin, L., Yu, F., Estrin, D., Govindan, R., Shenker, S.: GHT: A geographic hash table for data-centric storage in sensornets. In: Proc. of the 1st ACM International Workshop on Wireless Sensor Networks and Applications (WSNA) (September 2002)
13. Karp, B., Kung, H.T.: GPSR: greedy perimeter stateless routing for wireless networks. In: Proc. ACM/IEEE MobiCom, pp. 243–254 (August 2000)

14. Levis, P., Madden, S., Gay, D., Polastre, J., Szewczyk, R., Woo, A., Brewer, E., Culler, D.: The emergence of networking abstractions and techniques in TinyOS. In: NSDI 2004 (2004)
15. Welsh, M., Mainland, G.: Programming sensor networks using abstract regions. In: NSDI 2004 (2004)
16. Whitehouse, K., Sharp, C., Brewer, E., Culler, D.: Hood: a neighborhood abstraction for sensor networks. In: MobiSys 2004 (2004)
17. Intanagonwiwat, C., Govindan, R., Estrin, D.: Directed diffusion: A scalable and robust communication paradigm for sensor networks. In: Proceedings of the Sixth Annual International Conference on Mobile Computing and Networking (MobiCOM 2000), Boston, Massachussetts (2000)
18. Heidemann, J., Silva, F., Intanagonwiwat, C., Govindan, R., Estrin, D., Ganesan, D.: Building efficient wireless sensor networks with low-level naming. In: Proceedings of the Symposium on Operating Systems Principles, Lake Louise, Banff, Canada (2001)
19. Ganesan, D., Greenstein, B., Perelyubskiy, D., Estrin, D., Heidemann, J.: An evaluation of multi-resolution storage for sensor networks. In: Proceedings of the ACM SenSys Conference (2003)
20. Madden, S., Franklin, M.J., Hellerstein, J., Hong, W.: TAG: A tiny aggregation service for ad-hoc sensor networks. In: Proceedings of the 5th USENIX Symposium on Operating Systems Design and Implementation (OSDI 2002), Boston, Massachusetts (December 2002)
21. Yao, Y., Gehrke, J.: Query processing in sensor networks. In: Proc. of the First Biennial Conference on Innovative Data Systems Research (CIDR 2003) (January 2003)
22. Coman, A., Nascimento, M., Sander, J.: A framework for spatio-temporal query processing over wireless sensor networks. In: 2nd International VLDB Workshop on Data Management for Sensor Networks (2005)
23. Obraczka, K., Manduchi, R., Garcia-Luna-Aveces, J.: Managing the information flow in visual sensor networks. In: Proceedings of the 5th. International Symposium on Wireless Personal Multimedia Communications (October 2002)
24. Wolf, W., Ozer, B., Lv, T.: Smart cameras for embedded systems. IEEE Computer 35(9), 48–53 (2002)
25. Trivedi, M., Mikic, I., Bhonsle, S.: Active Camera Networks and Semantic Event Databases for Intelligent Environments. In: Proceedings of the IEEE Workshop on Human Modeling, Analysis and Synthesis, Hilton Head, South Carolina (June 2000)
26. Hampapur, A., Brown, L., Connell, J., Pankanti, S., Senior, A., Y-L, T.: Smart surveillance: Applications, technologies and implications. In: Proceedings of the IEEE Pacific-Rim Conference On Multimedia, Singapore (December 2003)
27. Esteve, M., Palau, C., Catarci, T.: A Flexible Video Streaming System for Urban Traffic Control. IEEE Multimedia 13(1), 78–83 (2006)
28. ACM 2nd International Workshop on Video Surveillance & Sensor Networks
29. Wilburn, B., Smulski, M., Kelin Lee, H.H., Horowitz, M.: The light field video camera. In: SPIE Electronic Imaging 2002, Media Processors, Conference Proceedings (2002)
30. Wilburn, B., Joshi, N., Vaish, V., Levoy, M., Horowitz, M.: High speed videography using a dense camera array. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2004), pp. 294–301 (2004)

31. Lacey, A.J., Pinitkarn, N., Thacker, N.A.: An Evaluation of the Performance of RANSAC Algorithms for Stereo Camera Calibration. In: Proceedings of The Eleventh British Machine Vision Conference (September 2000)
32. Shenker, S., Ratnasamy, S., Karp, B., Govindan, R., Estrin, D.: Data-centric storage in sensornets. In: Proc. 1st Workshop on Hot Topics in Networking (HotNets-I) (October 2002)
33. Priyantha, N., Chakraborty, A., Balakrishnan, H.: The Cricket location-support system. In: Proc. ACM/IEEE MobiCom (August 2000)
34. Wolfson, H.J., Rigoutsos, I.: Geometric hashing: An overview. In: IEEE Computational Science and Engineering, pp. 10–21 (October-December 1997)
35. Jannotti, J., Mao, J.: Distributed calibration of smart cameras. In: Proc. Workshop on Distributed Smart Cameras (DSC 2006) (2006)
36. Guéziec, A., Taubin, G., Horn, B., Lazarus, F.: A framework for streaming geometry in vrml. IEEE Computer Graphics and Applications, 68–78 (March/April 1999)
37. Guéziec, A., Taubin, G., Lazarus, F., Horn, W.: Converting sets of polygons to manifold surfaces by cutting and stitching. In: IEEE Visualization 1998 Conference Proceedings, pp. 383–390 (October 1998)
38. Guéziec, A., Bossen, F., Taubin, G., Silva, C.: Efficient compression of non-manifold meshes. In: IEEE Visualization 1999 Conference Proceedings (October 1999)
39. Guéziec, A., Taubin, G., Lazarus, F., Horn, W.: Simplicial maps for progressive transmission of polygonal surfaces. In: VRML 1998. ACM Press, New York (1998)
40. Taubin, G.: A signal processing approach to fair surface design. In: Siggraph 1995 Conference Proceedings, pp. 351–358 (August 1995)
41. Taubin, G., Horn, W., Lazarus, F., Rossignac, J.: Geometric Coding and VRML. Proceedings of the IEEE 86(6), 1228–1243 (1998)
42. Taubin, G., Guéziec, A., Horn, W., Lazarus, F.: Progressive forest split compression. In: Siggraph 1998 Conference Proceedings, pp. 123–132 (July 1998)
43. Taubin, G., Rossignac, J.: Geometry compression through topological surgery. ACM Transactions on Graphics 17(2), 84–115 (1998)
44. Balmelli, L., Taubin, G., Bernardini, F.: Space-Optimized Texture Maps. Computer Graphics Forum 21(3) (September 2002)