SHORT PAPER: Data-Centric Visual Sensor Networks for 3D Sensing *

Mert Akdere Uğur Çetintemel Daniel Crispell John Jannotti Jie Mao Gabriel Taubin
Brown University
{makdere,ugur,dcrispel,jj,taubin,jmao}@cs.brown.edu

ABSTRACT

Visual Sensor Networks (VSNs) represent a qualitative leap in functionality over existing sensornets. Cooperating networks of cameras could reconstruct features in three dimensions, produce images from novel viewpoints, match trajectories or objects against known patterns, or combine these tasks to provide a flexible monitoring system.

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 will process video data in real time, extracting features and 3D geometry from the raw images of cooperating cameras. These results will be stored and processed in the network, near their origin. New content-routing techniques will allow cameras to find common features—critical for calibration, search, and tracking. A novel query mechanism will 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. Each camera will perform local image processing, and then cooperate to perform higher-level tasks, such as calibration, view combination, object detection, and tracking.

Today, the largest camera networks are analog systems used for video surveillance. These systems consist of many cameras and capture a huge volume of data. The average casino has 2000-3000 cameras operating continuously. These systems require a small army of security personnel to monitor the video feeds. Smart event detection based on higher level analysis of the image data can help the security personnel manage this deluge of data.

Combining information from multiple cameras, trajectories of individuals can be computed, and suspicious behaviors can be identified. However, large camera systems must avoid streaming all video data to a central server in order to scale. Instead, we are developing distributed techniques that allow for the fusion of information across cameras.

*This work is supported in part by the NSF under grant IIS-0448284.

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 in use at such an airport today, augmented with thousands of additional static cameras to gain greater coverage, and with many more mobile cameras attached to airport personnel.

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 human suspicion, 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.

Virtual Views. With sufficient coverage, a VSN can generate virtual views—images that are the result of combining data from multiple cameras in order to generate views from a new angle or with greater resolution.

For each pixel in an image, there is a unique directed straight line, or *ray*, which leaves the camera. The mapping from 3D rays to colors is known in computer graphics as the *light field*. Virtual views are generated by interpolating sample values in the light field. In a VSN, light field samples come from individual images stored in a distributed fashion in or near the smart cameras where they were observed. When a user issues a request for a virtual view, the request must be routed to the smart cameras containing relevant pixels, with multiple cameras possibly contributing to different parts of the virtual image. By super-sampling

the light field across multiple overlapping images, *super-resolution* images may be generated with greater clarity than any of the individual image frames.

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 blobs detected in different images requires feature detection and matching. 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. The feature data interchanged between cameras is not large, but the complexity of feature matching is, in principle, quadratic in the number of cameras. Recent contributions [4, 3] that address this problem seem to be appropriate for VSNs.

Storage. In a large-scale VSN, it is neither efficient nor practical to continuously stream acquired data to external locations for persistent, long-term storage. To answer historical questions, therefore, VSN nodes need to have local storage. Local storage also facilitates in-network processing, which potentially increases the overall efficiency and effectiveness of the VSN.

To make the best use of available storage, A VSN should first eliminate redundant data through coordinated storage—close nodes can communicate, identify common data, and avoid storing it at multiple locations. In addition, nodes should store data selectively and potentially at varying resolutions. It is often the case that more recent data is more valuable than older data, so data could be aged out through increasingly lossy compression. Similarly, certain image regions (such as the background or static objects) could be entirely discarded or compressed more than other regions.

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 acquired by a VSN is more 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 of 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, in fact, 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 [2] which builds a more traditional routing tree.

Space-Time Database Abstraction. Our proposal contains two key components that simplify the development of 3D sensornet applications. First, we use *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. Second, we rely on a predicate language for specifying space-time *feature patterns* for search and tracking of complex objects and activities easily.

2. NETWORK PRIMITIVES

In existing sensor networks, the need to route requests to the sensors best able to make a specific observation has been met by localization to determine sensor position, and routing primitives that operate on position. In visual sensor networks, the localization must be generalized to include orientation, and routing must be generalized to account for long range sensing.

2.1 Data-Directed Localization

In VSNS, even small localization errors may be unacceptable when a distant object's location is estimated. Existing techniques, such as GPS and Cricket, are too coarse to match image data from independent cameras. We propose *data-directed localization* in which smart cameras localize with respect to each other based on shared image data. Sensor nodes detect local features and then cooperate to find common features observed by multiple cameras, allowing the nodes to orient themselves in a shared coordinate system. Additional cameras may orient themselves in this system by finding features in the shared space.

Data-directed localization requires that sensors find similar features in nearby cameras. Unfortunately, low-level two-dimensional features are very difficult to match between uncalibrated cameras. Instead, we advocate smart cameras with two image sensors. Using two sensors with a known (short) baseline allows for the local recognition of 3D features from the 2D images. 3D features drastically reduces the search space when considering inter-node matches.

We have prototyped this approach using several *camera pods*. Each camera pod includes four rigidly mounted network cameras capable of small baseline feature matching and stereo reconstruction. 3D features locations were estimated

and matched between pods. Using at least three 3D feature correspondences, a rotation and translation was calculated to bring the seperate pods into a common reference frame.

2.2 Feature-oriented Search and Computation

Our prototype explored data-centric calibration, but performed inter-node feature matching in a centralized way. Real VSNs must find correspondences without centralization. We build on the idea of Geographic Hash Tables [7] to support decentralized inter-node matching. Features will be placed in the GHT, binned by geometric (*not* geographic) hashing [10]. Similar features will therefore be placed at the same location. Nodes with shared features can be notified, and a relative transform computed.

Inter-node feature matching is one case of a more generic VSN service—content-based search. GHTs operate by hashing a data item key to a geographic coordinate, and storing the item at the node closest to that coordinate.

Assuming data with a key corresponds to the needs of queries, retrieval is efficient. For example, suppose that cameras can detect and measure the heights of people they observe. They might store observations in the GHT keyed by those heights, binned into one inch increments. A query can find all individuals of a certain height by examining the hash location associated with the potential observation of such an individual.

Hashing Hints. Performance improvements can be obtained by using a hash function that creates locality among 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 will be inserted into the GHT with keys that include the rounded location of the observation, in order to facilitate queries on the location. Hashing each such key results in the storage of these observations arbitrarily throughout the sensornet, requiring our example query to collect values from several locations.

By widening the GHT's insertion and lookup interfaces to mark portions of the key as a geographic "hint", spatial locality can be preserved. The hash function is modified to set the high bits of the coordinate at which the data will be stored linearly, according to the hints. Thus, queries that access ranges over the hinted attributes will require data retrieval from a single geographic area.

Data-Centric Processing. The detection of high-level features, such as faces, is generally accomplished by detecting simpler features (eyes, noses) in a particular arrangement. We have seen that simple 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 facial observation in the GHT. The data included with this observation indicates that it is only a partial observation. A similar aggregator creates partial observations for eyes and noses. These second-level observations are inserted under the same name, and therefore make their way to the same location. When enough observations agree, a face has been detected. These operators, placed in the GHT to process values at their insertion point, are the natural computational analogue to data-centric routing and storage—data-centric processing.

2.3 Image Based Routing

Data-centric techniques are best suited for queries that operate on meta-data, and are intended to find features that may appear anywhere. When the focus is on retrieving data from a known location, *Image Based Routing* [2] is more appropriate. In IBR, smart cameras build a representation of their fields of view and pass these representations up through a routing tree. We have developed the "binmesh" which succinctly represents the views of many cameras in a single summary. Queries follow the binmeshes down the routing tree toward cameras that observe the target object.

3. DATA ACCESS AND QUERYING

3.1 Basic Model and Primitives

An important goal of our system is to simplify VSN application development. Any sophisticated VSN will require search and detection of objects, activities, and complex events based on images. Space-time point queries, range queries and content-based similarity (k-NN) will also be common. An extensible, high-level programming framework that enables the specification and monitoring of complex objects and activities of interest is a key requirement.

Multi-Level Data Representation. Our primary programming abstraction is a space-time 4D view of the underlying data, consisting of a 3D volume (*x,y,z*) representing geographic space and the fourth dimension t representing time. This abstraction captures the data produced by all the sensors in a sequence of time frames, where each frame is a 3D cube that provides a logical model of the world of interest. The abstraction allows users to query the system based on spatial attributes on a combination of live and historical data. This view is *virtual* with the implication that the execution of a query involves accessing the distributed base data (*e.g.*, as in Cougar [12] and TinyDB [6]).

Our framework also includes a *raw data layer* and a *feature layer*. The raw data layer continuously acquires and stores camera data. The cube layer transforms raw sensor data into the (virtual) cube abstraction. The feature layer consists of cascadable space-time views defined over the cube layer.

Basic Data Access Methods. The basic image data access and query interface is a linear, SQL-like notation that facilitates declarative queries over the cube. Consider the following query notation:

```
SELECT from CUBE
WHERE location = bbox
WHEN time = interval
VIEWPOINT = vp
WITH RESOLUTION k
SAVE AS VIEW name
```

This continuous query can be used to select a volume of space specified by its bounding box and ask for an image stream that corresponds to the target volume as observed from a specific viewpoint. The query may specify a time interval, which may refer to the past (clearly not all historical queries can be answered with limited storage). The desired temporal resolution of the stream can be also be specified. Finally, the query may be saved as a named view.

Space-Time Features. Our system achieves extensibility by allowing users to specify complex *spatial* and *temporal* features through the composition of base features. A base *feature extractor* is defined by a user defined function (UDF) that analyzes an input image and returns a set of vectors, one for each feature detected. A feature vector contains a set of named attributes (*e.g.*, color, bounding box, *etc.*). One attribute is a probability value indicating the confidence about the occurrence of the feature. An input threshold on this value is used to discard unlikely candidates. More complex spatial features are defined based on relationships of lower-level features over space as:

`DEFINE` feature name `WITH` $f_1, f_2, ..., f_n$ `WHERE` $p_1, p_2, ..., p_m$ where $f_1, f_2, ..., f_n$ are the simpler features to be composed and $p_1, p_2, ..., p_m$ are the predicates that define the relationship among these features. The relationships are expressed using a small set of spatial predicates similar to those in Allen's algebra [1], *e.g.*, a nose is *above* the mouth.

Temporal features will be defined in a similar manner although, in this case, one is interested in the variation of the spatial orientation of a feature over time. For example, the activity of "moving" can be expressed as a specific feature changing its location. We are exploring techniques from moving objects and motion database research to represent and express complex movement patterns [11, 5].

Multiple 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 with a temporal predicate that checks whether that person is moving faster than some threshold. To compose complex space-time patterns, we use an extended event composition algebra (consisting of disjunctions, conjunctions, sequences, and negations) which allows for space and time constraints to be associated with event instances. Another novel feature of the algebra is that it takes into account uncertainty in feature detection using probabilistic models.

We note that conceptually similar feature-driven querying models have been proposed earlier (*e.g.*, in immersidata management [9] and visual surveillance systems [8]). The new challenge we address is to design a practical model that effectively handles uncertainty and lends itself to efficient distributed implementation.

3.2 Query Execution

Once the user submits a query to the system, the query will be translated into an execution plan. The planning phase decides, based on the query specification, which routing indexes (image-based, spatial, temporal, feature-based, or a combination) and feature detectors to use. The query plan will then be distributed and executed collectively by the appropriate nodes in a distributed fashion, after which results are streamed back to the user.

One important goal of our execution strategy is to avoid shipping raw images as much as possible. The basic operation model we envision is one where feature-based queries are used to detect interesting events, which will then steer the attention of human operators to zoom in on the activity by requesting images using cube layer queries.

When a cube-layer query is specified, the system uses the image-based routing scheme (Section 2.3) to route the request to the cameras that can collectively provide the best view of the specified sub-cube. When a query refers to a feature, the corresponding base and higher-level extractors will be sent to the appropriate nodes and executed using data-centric processing (Section 2.2). Here, the GHT may also act as a spatial filter—if a feature-based query specifies a target region, then the base feature extractors need only be executed in that region.

We assume best-effort semantics for query execution; for example, if a specific viewpoint cannot be presented due to lack of data, the system might offer an alternative, similar viewpoint for which data is available. Likewise, the system may have to operate below the target resolution when bandwidth is scarce or may not answer historical queries if the data is no longer available. We will rely on adaptive compression of image streams (*e.g.*, using wavelets) to ameliorate both problems.

4. 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 the initial problem of self-calibration, through feature and image retrieval, to an expressive and efficient query language for application interaction.

5. REFERENCES

[1] J. Allen. Maintaining knowledge about temporal intervals. *Communications of ACM*, 26(11):832–843, 1983.

[2] D. Crispell, G. Taubin, and J. Jannotti. Image based routing for image based rendering. In *Proceedings 6th Workshop on Omnidirectional Vision, Camera Networks, and Non-classical Cameras (OMNIVIS 2005)*, Oct. 2005.

[3] K. Grauman and D. T. Efficient image matching with distributions of local invariant features. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition*, San Diego, CA, June 2005.

[4] K. Grauman and D. T. The Pyramid Match Kernel: Discriminative Classification with Sets of Image Features. In *Proceedings of the IEEE International Conference on Computer Vision*, Beijing, China, Oct. 2005.

[5] Keogh, Palpanas, Zordan, Gunopulos, and Cardle. Indexing large human-motion databases. In *Proc. of the 30th International Conference on Very Large Data Bases*, Aug.

[6] S. Madden, M. J. Franklin, J. Hellerstein, and W. Hong. TAG: A tiny aggregation service for ad-hoc sensor networks. In *Proceedings of the 5th USENIX Symposium on Operating Systems Design and Implementation (OSDI '02)*, Boston, Massachusetts, Dec. 2002.

[7] S. Ratnasamy, B. Karp, L. Yin, F. Yu, D. Estrin, R. Govindan, and S. Shenker. 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)*, Sept. 2002.

[8] Saykol, Gudukbay, and Ulusoy. A database model for querying visual surveillance by integrating semantic and low-level features. In *Proc. of 11th International Workshop on Multimedia Information Systems*, 2005.

[9] C. Shahabi. Aims: An immersidata management system. In *Proc. of the First Biennial Conference on Innovative Data Systems Research (CIDR'03)*, 2003.

[10] H. J. Wolfson and I. Rigoutsos. Geometric hashing: An overview. *IEEE Computational Science and Engineering*, pages 10–21, October-December 1997.

[11] O. Wolfson, A. P. Sistla, B. Xu, J. Zhou, and S. Chamberlain. Domino: Databases for moving objects tracking. In *Proc. of the 1999 ACM SIGMOD International Conference on Management of Data*.

[12] Y. Yao and J. Gehrke. Query processing in sensor networks. In *Proc. of the First Biennial Conference on Innovative Data Systems Research (CIDR'03)*, Jan. 2003.