VISNET: A DISTRIBUTED VISION TESTBED

Michael Quinn *Raghuraman Mudumbai* *Thomas Kuo* *Zefeng Ni* *Carter De Leo* *B.S. Manjunath*

University of California - Santa Barbara
Department of Electrical and Computer Engineering
Santa Barbara, CA 93106

ABSTRACT

We introduce UCSB's Visual Sensor Network (VISNET) and discuss current research being conducted with the system. VISNET is a ten-node experimental camera network at UCSB used for various vision-related research. The mission of VISNET is to provide an easy-to-use multi-node camera network to the vision research community at UCSB. This paper briefly discusses design and setup considerations before discussing current research. Current research includes operation visualization, camera network calibration, tracked object modeling, and multiple object / multiple camera tracking.

Index Terms— Camera Networks, Smart Cameras, Tracking, Sensor Networks

1. INTRODUCTION

Rapid advances in technology have changed the goal of visual surveillance from building systems using only a few powerful cameras to building systems deploying many cheap cameras. Given a large camera network (e.g. hundreds of camera nodes), a typical architecture with one central server collecting the video streams and performing all of the analysis would fail due to communication and computation constraints. However, as the capabilities of a camera's on-board processing advance, distributed solutions have become feasible. The question is how to make such a distributed system scalable without compromising its performance.

In this paper, we introduce our distributed vision testbed – Visual Sensor Network (VISNET). Our setup creates flexibility, since the processors we use allow us to consider algorithms first, and processing constraints later, while the visualization system allows us to see our results quickly. In this system, we take advantage of the overlapping views of multiple cameras to contribute three algorithms for different areas of computer vision: (1) a distributed camera calibration algorithm, (2) a simple human appearance model, and (3) a multiple human tracking scheme.

Distributed Camera Calibration

The problem of camera calibration [1, 2, 3] is a well-studied problem in the vision literature. Indeed calibration is used in any application that studies objects in 3D space through 2D images. For instance, work on gesture recognition [4] and virtual reality [5] all use calibration techniques based on the camera projection model. However, surprisingly little attention has been paid to the automatic calibration of a distributed large camera network, especially under the constraint of bandwidth and computation. We propose a simple, distributed calibration method that uses a moving planar pattern and automatically calibrates the network in a way that is much less complicated than classical nonlinear estimation [6] with comparable results.

Simple Multiview Human Model

With the calibrated cameras, one task we approach is to create a human model for data association across cameras and identification during tracking. Much of the work in this area has been with single cameras, and either 2D models or inferred 3D models. Zhou and Hoang [7], for example, uses position, velocity, color histograms, and the number of pixels to represent tracked people. Work in multicamera networks has expanded upon the single camera models. Wu and Aghajan [8] fit colored ellipses to body segments to build a 3D articulated human model. Our approach is to instead to eschew the more complicated models and build a simple, multiview human model that can be updated at a central location.

Multiple Person Tracker

Adding a human model to camera calibration allows us to address tracking, a popular topic in computer vision. Challenges to good tracking include tracking multiple people, especially in a crowd, and tracking in the presence of occlusions. With a single camera, Rasmussen and Hager [9] uses a joint probability data association filter (JPDAF) to re-associate objects after occlusion. With multiple cameras, Cai and Aggarwal [10] and Khan et al.[11] use the camera with the best view and hand off the tracking to neighboring cameras when the target leaves the field of view. Ercan et al.[12] also use multiple cameras and a particle filter to handle static and moving occluders. In this paper, we present a method that avoids complicated filters like JPDAF and particle filters. Instead,

978-1-4244-2665-2/08/$25.00

taking advantage of the overlapping cameras, our method uses only a Kalman filter and transmission of predicted 3D position to the nodes to deal with multiple objects and occlusions.

To minimize data communication in VISNET, all processing related to images are done locally. This is similar to the system presented by Xu et al. [13] for tracking people in sports applications. They use a dedicated feature server to process video from each camera and a multi-tracker module that only needs to merge information from individual trackers using a nearest neighbor method. Communication and synchronization is achieved through a "request-response" pattern, invoked by the multi-tracker. In this paper, we use a different mechanism where all camera nodes and central nodes simply broadcast their tracking results and synchronization is achieved through timestamps. This removes the necessity for a central node to coordinate all the message passing and allows the system to scale easily.

Section 2 details the hardware and setup used in VISNET and Section 3 describes our main interface to the system. In Section 4, we describe our camera calibration scheme. Section 5 briefly overviews the simple 3D object model. Finally, Section 6 describes our object tracker with results.

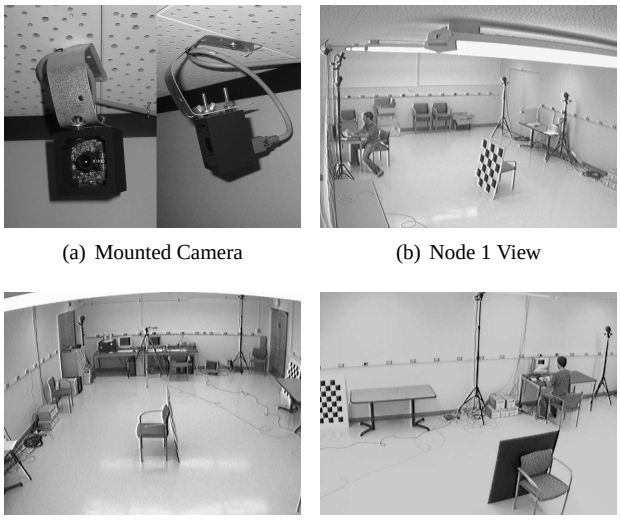
2. CAMERA NETWORK DESIGN

The driving force behind the VISNET system is camera sensor network research. We envision a network of camera-enabled autonomous nodes which could be easily configured for a variety of applications. As our interests lie in software and algorithms rather than hardware design and configuration, the system consists of off-the-shelf hardware.

Each of the ten VISNET nodes consists of a PC with an attached IEEE 1394 camera. This setup provides us with the flexibility to test algorithms. For simplicity, the nodes are networked via wired Ethernet. An eleventh node serves as a central tracker and is also used for application control and visualization. All nodes run the Ubuntu Linux distribution and applications are developed using open source tools.

The room is 10 meters long by 6 meters wide by 2.8 meters high, and the ten cameras are mounted approximately 150 mm from the ceiling pointed slightly downward. Fig. 1 shows a mounted camera along with three typical camera views.

In order for cameras to collaborate in tracking and other applications, it is important that data correspond to the same time instant. Due to the difficulty of synchronizing the capturing and processing of video, we employ timestamps to ensure data coherence. However, this approach requires the timestamps produced at each node are accurate. While exact synchronization is not required and likely not possible, the timing errors should be small compared to the timescales of the target motion. To this end, we employ the Network Timing Protocol (NTP) [14]. NTP synchronizes the camera nodes' clocks with the central node. Our timing error using NTP are on the order of 4 ms which is sufficient for tracking people in the VISNET system.



(a) Mounted Camera (b) Node 1 View

(c) Node 5 View (d) Node 8 View

Fig. 1. VISNET Camera and Views

3. SYSTEM VISUALIZATION AND CONTROL

To visualize and control the operation of VISNET, we have built a GUI front end for the system. The visualization node communicates with the central node using a custom message format. When the front end first establishes its connection to the tracking server, it requests a description of the topology of the camera network and the parameters for each camera. The front end then draws a 3D representation of each camera in the network using OpenGL, as shown in Fig. 2.

From this view, the user can select an individual node to see live video from the corresponding camera. Specifically, the interface periodically requests video frames from the camera node, which encodes the frames using MPEG4 and transmits them to the visualization.

The interface also periodically requests the locations of objects being tracked by the central server. After receiving the 3D and 2D locations of each object in the selected camera's viewpoint, the front end draws markers along with the received frame.

4. CAMERA NETWORK CALIBRATION

The goal of camera network calibration is to determine the nodes' positions and orientations with respect either to each other or to some world reference. In our system, the calibration is performed in real time using a planar chessboard calibration pattern since planar patterns are easy and cheap to build. During the calibration process, the chessboard is

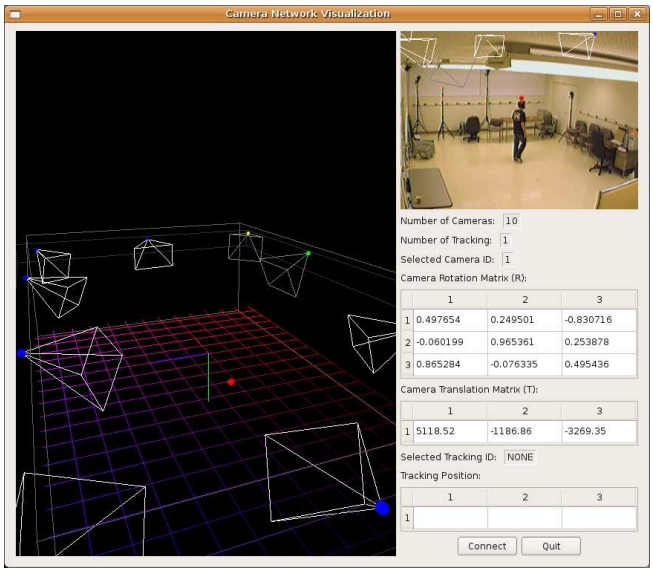


Fig. 2. Visualization Front End

moved through the network. Calibration based on planar patterns has been proposed in literature for single camera calibration, such as [1, 2]. However, how to use it to calibrate a large network has not been paid much attention. The main challenge is how to efficiently process thousands of planar patterns captured by the cameras over the time so that the network can be scaled easily. Here we propose a distributed calibration method, aiming to minimize communication and computation, without comprise to the performance.

When the chessboard moves in the network, each camera node i actively seeks the chessboard pattern. If the chessboard is detected at time t, the node performs typical extrinsic calibration as detailed in [1], i.e., estimate its own orientation $R_i(t)$ and position $T_i(t)$ with respect to the chessboard. $R_i(t)$ and $T_i(t)$ are stored at the camera node together with the timestamps t, which are then broadcasted to all the other nodes including the central control node.

When two nodes i and j detect the chessboard at about the same time (by comparing timestamps), their relative extrinsic parameters $\{\mathbf{R}_{ij}(t), \mathbf{T}_{ij}(t)\}$ can be easily computed. When this calibration between two nodes is performed multiple times, the mean relative parameters are computed. In this way, over time, the *Vision Graph* [15] is built, showing the successful relative calibration between nodes. A typical vision graph for VISNET is shown in Fig. 3.

Once sufficient internodal connections are calculated, the chessboard is placed in a world origin position (in our case, the center of the room) and those cameras which sense the board compute their extrinsic parameters with respect to it. The remaining nodes' *absolute* extrinsic parameters are then computed by tracing the path along the vision graph back to the world origin.

With the proposed method, the entire calibration process

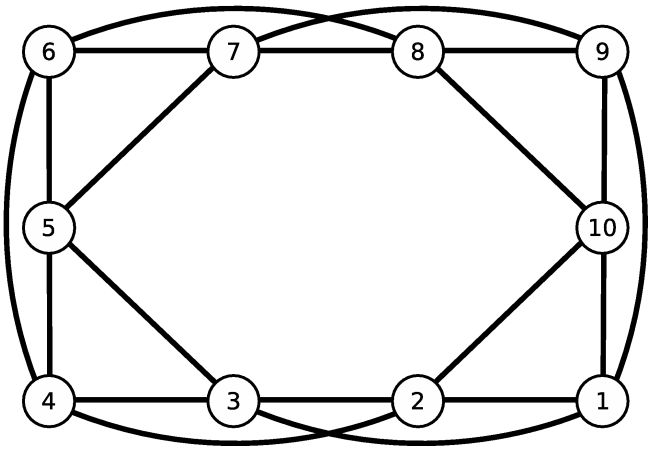


Fig. 3. VISNET Vision Graph - A connection between nodes indicates a shared field of view

is done in a distributed fashion, which makes the it fully scalable to a large network.

1. Each camera node actively detects the chessboard and passively computes its relative orientations and positions to neighboring cameras. The calibration does not need any control from the central node, which makes it feasible for a network without central control.

2. The communication among the nodes is minimized by broadcasting $\{R_i(t), T_i(t)\}$ instead of actual images or 2D pixel positions of the detected chessboard pattern.

3. Image processing (the detection of the chessboard) and the estimation of $\{R_i(t), T_i(t)\}$ are performed at the individual nodes. In addition, the relative extrinsic parameters between cameras $\{\mathbf{R}_{ij}, \mathbf{T}_{ij}\}$ are simply the average of $\{\mathbf{R}_{ij}(t), \mathbf{T}_{ij}(t)\}$ over time. Compared with a method that estimates extrinsic parameters directly from all the 2D pixel measurements from the chessboard (e.g. bundle adjustment [6]), our method requires much less complex computation. This feature could be crucial when camera nodes have only limited computing power and internal memory (e.g. a small smart camera).

To evaluate our distributed calibration method, we use the calibration results to locate a set of sparsely-spaced points in 3D and compare the localization results with ground truth. The average error is about 2cm in the current setup of 10 cameras. We also tried a different calibration method which collects the 2D pixel positions of the chessboard from all the camera nodes over the entire calibration process and estimates the extrinsic parameters directly using a global bundle adjustment method [6]. Using results from this global calibration method, we locate the same set of points in 3D. The same average localization error is observed. Fig. 4 shows the localization error for both methods. We can see that the global

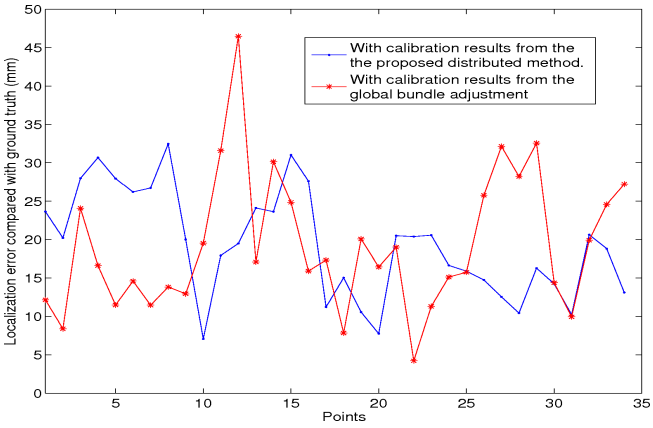


Fig. 4. 3D localization error

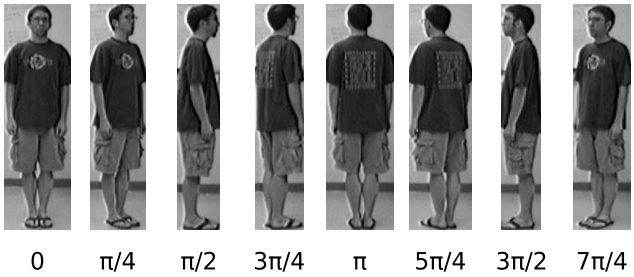


Fig. 5. Multiview Human Model

calibration method does not provide much improvement for the 3D localization compared to our distributed method.

5. LEARNED MULTIVIEW HUMAN MODELING

Given that the cameras are calibrated, we turn our attention to tracking people. However, it is desirable when tracking to assign each a unique signature which can be used to differentiate people when complex situations such as occlusion or interaction occur. Such a signature is also useful when a system is composed of two or more disjoint subsystems as it can enable the subsystems to share information about people and events in each area. One common method uses color features of the tracked region, i.e. the person's silhouette. The VISNET system utilizes a multiple-view color-based appearance model for tracking. The model is composed of eight views as shown in Fig. 5.

Each 2D view of a person is segmented into head, torso, and legs sections and each section is represented by color information in the CIE*Luv color space. Each region is modeled as a Gaussian mixture model (GMM) using the Expectation Maximization (EM) algorithm. In addition to being used for nodal tracking, the resulting mixture model is transmitted to the central tracker for inclusion into the global appearance model for each tracked person. As a person is tracked over time, the corresponding global model is updated using inputs from all sensing camera nodes. While we currently assume no occlusions during the training period, the learning algorithm could be easily adapted to account for such issues. Fig. 6 shows an example of a typical human GMM appearance model.

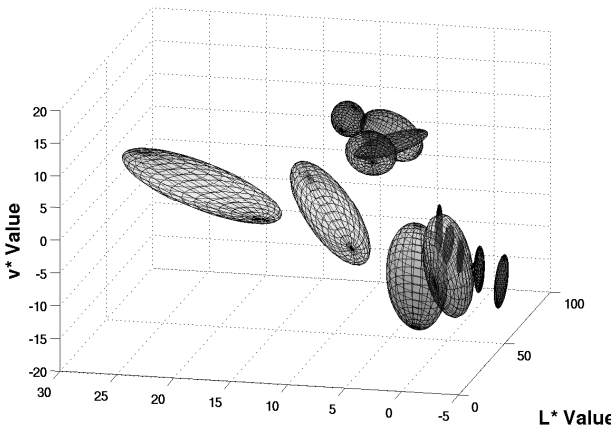


Fig. 6. GMM Human Appearance Model Example

6. MULTIPLE HUMAN TRACKING

VISNET takes advantage of the calibration and the learned human model to track people within the space. Fig. 7 and 8 show the flow chart for the nodal and central processes, respectively. At each node, a Kalman filter estimates the position of each person visible in a camera. The central server also has a Kalman filter that tracks the 3D position of every person in the space. To track multiple people across occlusions, we assume that any person is visible to at least two cameras, and then pass the 3D localization back to the nodes. As we show experimentally, this creates better results than Kalman filtering without feedback, yet is not as complicated as JPDAF or particle filters.

6.1. Nodal Tracking

In a camera network, the nodes initiate the process of tracking people through detection. The camera node in our case handles most of issues with occlusions and multiple objects with the help of the 3D positions of the existing people. The flowchart in Fig. 7 shows the major blocks in the nodal process.

To detect people, VISNET extracts silhouettes, or masks, using background subtraction in the CIE*Luv colorspace. This colorspace reduces the effect of shadows and other illumination abnormalities with minimal computation. To extract a bounding box, we avoid the typical, noisy connected components analysis. Instead, in a method inspired by Ercan [12],

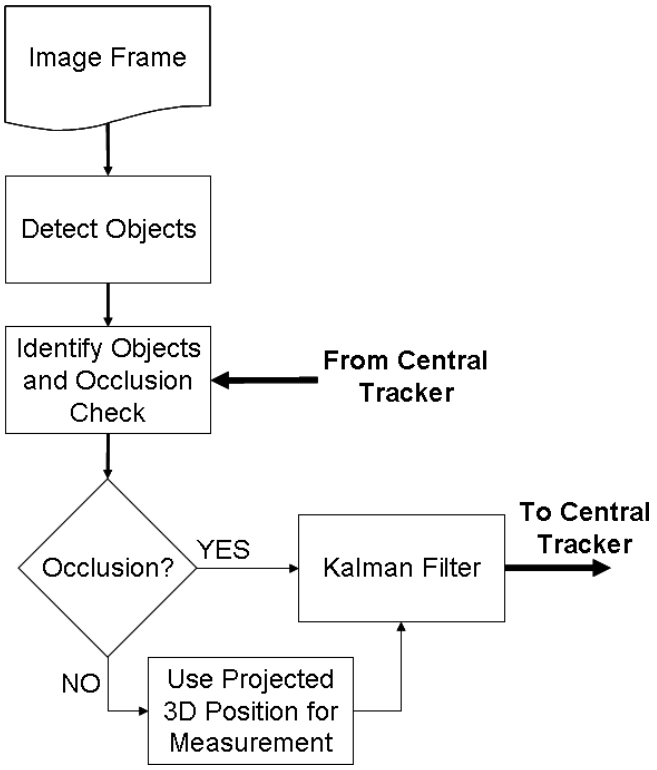


Fig. 7. Flowchart for tracking at camera node.

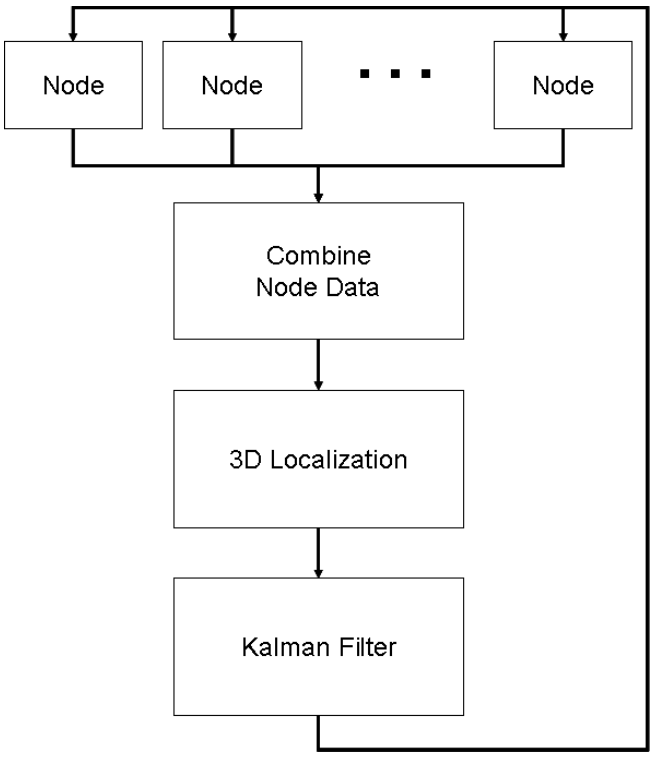


Fig. 8. Flowchart for tracking at central node.

the mask is, first, projected vertically in order to identify horizontal head position. Then, an edge detector runs vertically from the head location to find the head and feet, and horizontally to find the sides.

After detection, the node tries to identify the objects and check for occlusion. With a local Kalman filter, multiview model, and global predicted points, it identifies matches the observed person to existing objects in the system through nearest neighbor analysis and model comparison. Simultaneously, the global predictions detect occlusions that cause two masks to combine into one.

Now, the Kalman filter is ready for update. We use a typical Kalman filter with a state $\mathbf{x_k} = [\ x \quad \dot{x} \quad y \quad \dot{y}\ ]^\mathbf{T}$, where (x, y) represent the image coordinates and $(\dot{x}, \dot{y})$ represent the velocity, and a measurement of $\mathbf{z} = [\ x \quad y\ ]^\mathbf{T}$. Thus, our Kalman filter is:

$$\mathbf{x_k} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \mathbf{x_{k-1}} + \mathbf{v_k} \quad \mathbf{v_k} \sim \mathcal{N}(\mathbf{0}, \mathbf{Q_k}) \quad (1)$$

$$\mathbf{z_k} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \mathbf{x_k} + \mathbf{w_k} \quad \mathbf{w_k} \sim \mathcal{N}(\mathbf{0}, \mathbf{R_k}) \quad (2)$$

We vary z_k and R_k based on the occlusion.

$$\mathbf{z_k} = \begin{cases} \mathbf{P} \cdot \mathbf{Z_k} & \text{if object is occluded} \\ \mathbf{z_h} & \text{if object is not occluded} \end{cases} \quad (3)$$

where $\mathbf{P}$ is the projection matrix for that camera and $\mathbf{Z_k}$ is the 3D point for the person, and z_h is the head location observed by the camera.

$$\mathbf{R_k} = \begin{cases} 2\mathbf{R_h} & \text{if object is occluded} \\ \mathbf{R_h} & \text{if object is not occluded} \end{cases} \quad (4)$$

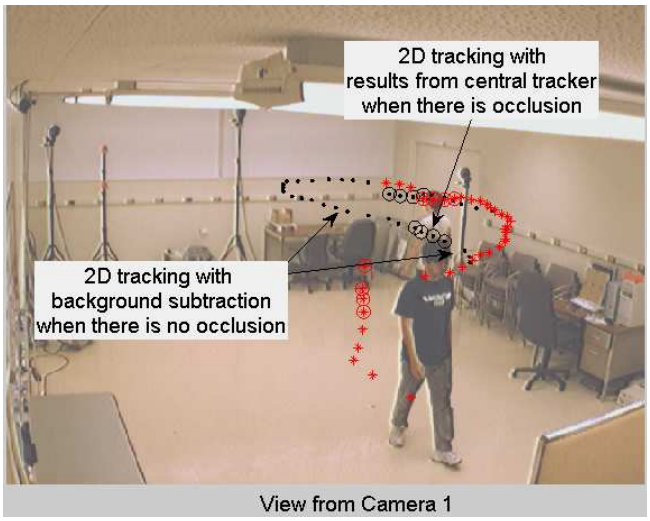
where $\mathbf{R_h}$ is measurement noise of the person, which is chosen heuristically.

Finally, the data is sent to the central tracker for further processing.

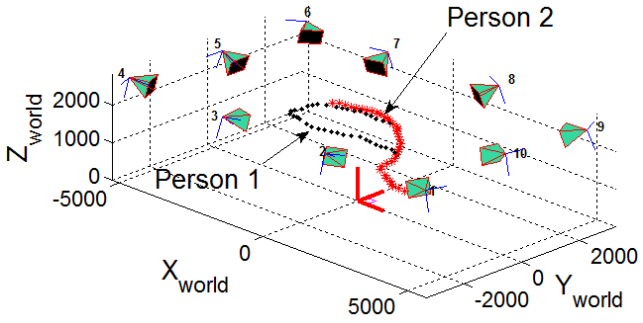
6.2. Centralized Tracking

The centralized tracking process receives inputs from all camera nodes within the network, combines and computes the 3D location for the Kalman filter, and sends the filter predictions to the nodes, as shown in Fig. 8.

The central tracker must first re-sort incoming data according to the identifies found by the nodes. Existing objects are trivially sorted. New objects, however, requires the central server to correspond points through other means and create a new multiview model, Sec. 5.

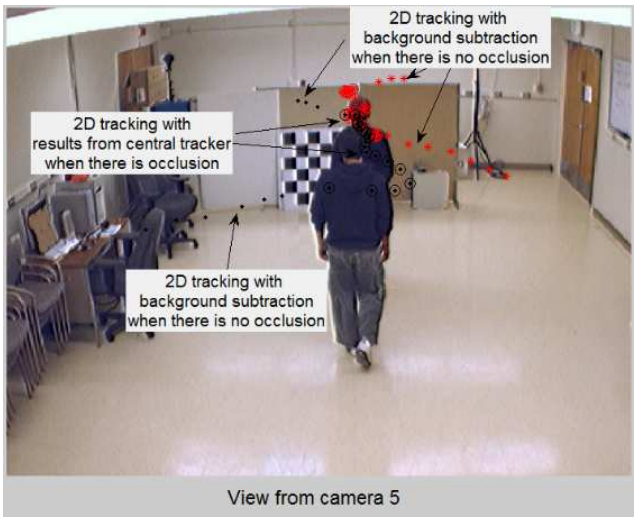


(a) 2D Track with feedback

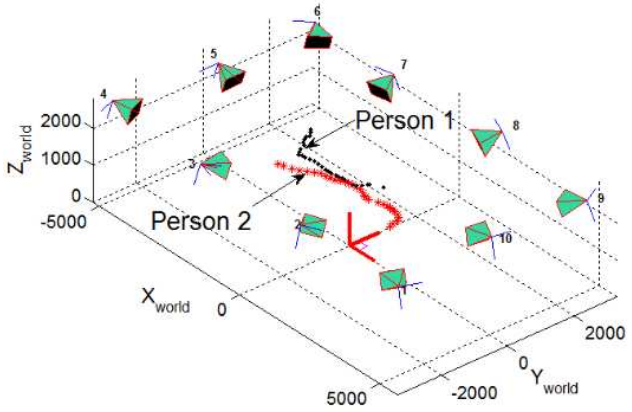


(b) 3D Track from all cameras

Fig. 9. Tracking results for looping paths.



(a) 2D Track with feedback



(b) 3D Track from all cameras

Fig. 10. Tracking results for crossing paths.

Then, the tracking process localizes the 3D with the nodal observations. Localization is computed by a least square estimate using only unoccluded observations. Occluded observations are based on the prediction and thus not necessary in our case.

Next, the tracker adds the 3D localization to the person's Kalman filter. The Kalman filter is very similar to the Kalman filter in Eq. 2, except that there are three coordinates: $\mathbf{X_k} = [\ X \quad \dot{X} \quad Y \quad \dot{Y} \quad Z \quad \dot{Z}\ ]^{\mathbf{T}}$, and $\mathbf{Z_k} = [\ X \quad Y \quad Z\ ]^{\mathbf{T}}$ where (X, Y, Z) and $(\dot{X}, \dot{Y}, \dot{Z})$ represent the 3D position and velocity respectively.

The tracking prediction is now sent to the nodes to start the next cycle.

6.3. Experimental Results

To test our tracking system, we ran tests on several scenarios collected within the data. We present here two representative scenarios of multiple object tracking with occlusion, that VISNET collected and processed.

In the first scenario, one person enters first and begins to walk around the room. A second person then enters, and also walks around the room in a different path. The two people often occlude each other for short periods when they pass each other. A 3D estimation of the path created by VISNET can be seen in Fig. 9(b).

The tracking results with feedback from camera 1 are shown in Fig. 9(a), and the localization result is shown in Fig. 9(b). In camera 1, the two targets are occluded once. During the first occlusion, no other cameras can see person 2; therefore, their position is updated by a Kalman filter prediction. After the occlusion is resolved, the result converges on the true track.

In the second scenario, we're mainly concerned with the view from camera 5, shown in Fig. 10(a). From this camera, two people start at the bottom on opposite corners, walk away from the camera and towards the middle, until they are walking in a single file line away from the camera. After a short

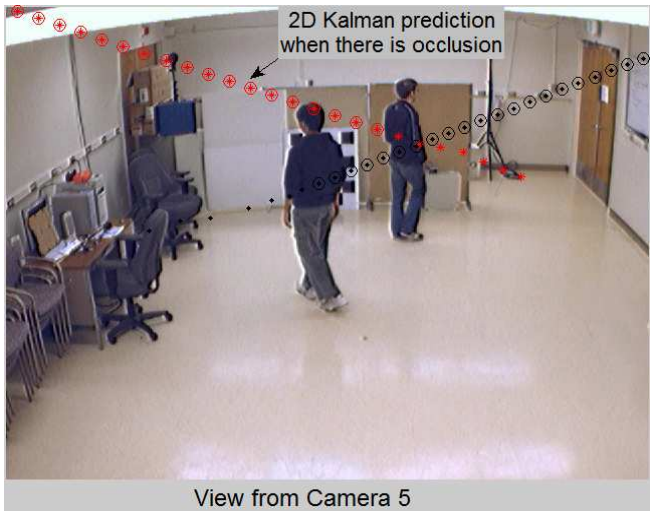


Fig. 11. Kalman filter track without feedback.

time, they separate to the same side of the camera view from which they originated. This path can be seen in Fig. 10(b).

The results with feedback in the main camera is shown in Fig. 10(a), and the 3D localization from all cameras is shown in Fig. 10(b). Within the node, the track remains fairly close, except for a slight deviation at the start of occlusion.

In a track created by a Kalman filter without feedback from the other nodes cameras, the predicted diverge at the start of occlusion, as shown in Fig. 11. These measurements are obviously incorrect and cannot be used in localization. Even if they are ignored during occlusion, after the occlusion, the Kalman filter will require time to converge to the new track. The tracker can either wait for convergence to send data or send bad data. The track collected with feedback is ready for use by the first non-occluded point.

Another possible solution for this scenario without feedback is to stop the 2D Kalman filter when a prolonged occlusion is detected, and then start a new one when measurements are again available. In this case, the data association would fail if a nearest neighbors condition were applied, and thus an appearance model or more complicated data association method would be required.

7. CONCLUSION AND FUTURE RESEARCH

This paper overviews UCSB's Visual Sensor Network (VISNET) which is a 10-node testbed for a smart camera network. On this system, we've developed novel algorithms in three areas: (1) a simple, pairwise calibration method that is less complex than bundle adjustment, but similar in performance; (2) a low-bandwidth multiview model using GMMs that can be updated at a central location; and (3) object tracking that uses feedback from the central tacker to the node to handle occlusions and multiple objects.

In the future, we will be expanding on all of these areas. In tracking, for example, we will be researching sending measurements from a block of time to reduce overall bandwidth, and, also, selection of the best set of cameras for tracking or dynamically setting the measurement and process covariances depending on the observation.

8. REFERENCES

[1] Zhengyou Zhang, "A flexible new technique for camera calibration," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 11, pp. 1330–1334, 2000.

[2] P. Sturm and S. Maybank, "On plane-based camera calibration: a general algorithm, singularities, applications," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 1999, pp. 432–437.

[3] Janne Heikkila and Olli Silven, "A four-step camera calibration procedure with implicit image correction," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR '97)*, Washington, DC, USA, 1997, pp. 1106–1112, IEEE Computer Society.

[4] C. Lin, W. Wolf T. Lv, and I. Ozer, "A peer-to-peer architecture for distributed real-time gesture recognition," in *Proceedings of the International Conference on Multimedia and Exhibition*, 2004.

[5] Simon Prince, Adrian David Cheok, Farzam Farbiz, ToddWilliamson, Nik Johnson, Mark Billinghurst, and Hirokazu Kato, "3d live: Real time captured content for mixed reality," in *ISMAR 02: Proceedings of the International Symposium on Mixed and Augmented Reality*, 2002.

[6] R. Hartley and A. Zisserman, *Multiple view Geometry in Computer Vision*, Cambridge University Press, 2003.

[7] Jianpeng Zhou and Jack Hoang, "Real time robust human detection and tracking system," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2005, pp. 149–156.

[8] Chen Wu and H. Aghajan, "Model-based human posture estimation for gesture analysis in an opportunistic fusion smart camera network," in *Proceedings of AVSS 2007*, 2007.

[9] Christopher Rasmussen and Gregory D. Hager, "Probabilistic data association methods for tracking complex visual objects," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 6, pp. 560–576, 2001.

[10] Q. Cai and J. K. Aggarwal, "Tracking human motion in structured environments using a distributed-camera system," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 21, no. 11, pp. 1241–1247, 1999.

[11] Sohaib Khan, Omar Javed, Zeeshan Rasheed, and Mubarak Shah, "Human tracking in multiple cameras," in *Proceeding of International Conference on Computer Vision*, Vancouver, Canada, 2001, pp. 331–336.

[12] Ali Ozer Ercan, Abbas El Gamal, and Leonidas J. Guibas, "Object tracking in the presence of occlusions via a camera network," in *IPSN '07: Proceedings of the 6th international conference on Information processing in sensor networks*, New York, NY, USA, 2007, pp. 509–518, ACM.

[13] M. Xu, J. Orwell, L. Lowey, and D. Thirde, "Architecture and algorithms for tracking football players with multiple cameras," *IEE Proceedings - Vision, Image, and Signal Processing*, vol. 152, no. 2, pp. 192–204, April 2005.

[14] David L. Mills, "The network time protocol (ntp) distribution," 2008.

[15] Dhanya Devarajan, Richard J. Radke, and Haeyong Chung, "Distributed metric calibration of ad hoc camera networks," *ACM Trans. Sen. Netw.*, vol. 2, no. 3, pp. 380–403, 2006.