GEOMETRIC TOOLS FOR MULTICAMERA SURVEILLANCE SYSTEMS

Eleanor G. Rieffel, Andreas Girgensohn, Don Kimber, Trista Chen, Qiong Liu

FX Palo Alto Laboratory
3400 Hillview Avenue
Palo Alto, CA 94304, USA
lastname@fxpal.com

ABSTRACT

Our analysis and visualization tools use 3D building geometry to support surveillance tasks. These tools are part of DOTS, our multicamera surveillance system; a system with over 20 cameras spread throughout the public spaces of our building. The geometric input to DOTS is a floor plan and information such as cubicle wall heights. From this input we construct a 3D model and an enhanced 2D floor plan that are the bases for more specific visualization and analysis tools. Foreground objects of interest can be placed within these models and dynamically updated in real time across camera views. Alternatively, a virtual first-person view suggests what a tracked person can see as she moves about. Interactive visualization tools support complex camera-placement tasks. Extrinsic camera calibration is supported both by visualizations of parameter adjustment results and by methods for establishing correspondences between image features and the 3D model.

Index Terms— surveillance, tracking, deployment support, camera placement, calibration, multiple views, geometric modeling, multicamera systems

1. INTRODUCTION

As cameras become cheaper and smaller, multiple camera systems are being used for a wide variety of applications including surveillance, business process analysis, and virtual experience of real places. Our DOTS system (Dynamic Object Tracking System), with over 20 cameras spread throughout the hallways and public spaces of our building, designed to aid workers in performing multicamera surveillance tasks, has been in constant use for over a year. DOTS makes it easy, for example, to track a person from camera view to camera view and map an image of a person to the corresponding 3D location of the person in the building. Figure 1 shows one of the standard views of the DOTS interface.

In the course of adding cameras and improving DOTS, we discovered that our tools were also useful for administering the system; in particular for exploring camera placement and performing calibration. Simple-to-use interactive visualization tools and analysis techniques aid interpretation of the spatial content of the surveillance views and the deployment and enhancement of the camera system. We describe tools that support spatial understanding, camera placement decisions, and camera calibration. This paper concentrates on that aspect of our system. See [1] for a detailed overview of the entire DOTS system, and [2, 3, 4] for applications of DOTS.

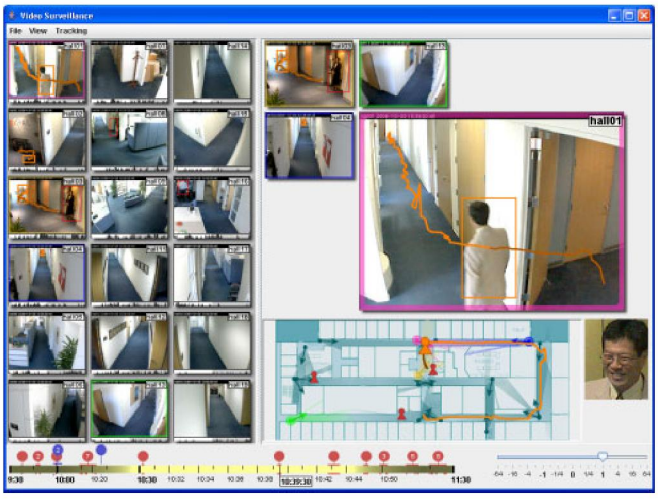


Fig. 1. DOTS system interface

Our work is novel in several aspects. First, few surveillance systems provide geographic context information. Our use of enhanced 2D floor plans and 3D environments helps users to stay oriented, and supports quick switching to an appropriate camera as security personnel follow people from camera to camera. Second, while a number of automatic planning techniques for camera placement exist, they support only restricted notions of good placement and most do not take 3D geometry into account. We provide users with interactive tools that visualize the consequences of manual camera placement. Our tools enable users to quickly place cameras and to check coverage based on criteria such as visibility at head height of a walking person, floor space covered by cameras, and live views of the cameras. Finally, our camera calibration tools show the alignment of the camera view with objects in

the 3D model. Users may either adjust camera parameters to improve the alignment or establish correspondence points that are used by our optimization algorithm.

In the next section, we discuss related work. Section 3 describes the DOTS system architecture and geometric input. The heart of the paper illustrates our tools for facilitating the understanding of camera views (Section 4) and for deploying and calibrating cameras (Section 5). We conclude with a discussion of future research directions.

2. RELATED WORK

Many commercial systems for video surveillance are available but only a few, such as VistaScape [5] and Praetorian [6], provide geographic context information. Both are geared at outside surveillance in settings such as utilities, transportation facilities, and military bases. Praetorean's Video Flashlight system enables users to explore a 3D model stitched together from multiple video camera views. VistaScape also has a spatial graphical user interface. Sebe et al. [7] use a virtual environment for an enhanced video surveillance experience. Our DOTS system is aimed at an indoor, office building setting, and makes significant use of 3D as well as 2D geometry, to aid in determining the location of tracked people, especially in cases of partial occlusion, and in representing what is visible to a camera.

For camera placement, our work is restricted to providing tools to enable a human to assess the camera placement within a space rather than providing an automatic optimal camera placement tool. Our work is therefore most closely related to that of State et al. [8] who, like us, are "uncomfortable handing off the design to an automated system" because they want to be able to "see and assess the trade-offs ourselves." Their work differs from ours in that it focuses on camera placement for 3D model reconstruction. While they mention that it could be useful for surveillance, it is hard to assess how well it would support surveillance tasks. Williams and Lee [9] also provide an interactive camera placement tool aimed at 3D reconstruction tasks. As we discuss in Section 5.1, it is at best tedious, and sometime impossible, to specify quality metrics that capture the varying importance of multiple desires and constraints over different locations in the space. Even when a suitable metric is available, determining the optimal camera placement is difficult; camera placement problems are closely related to the NP-hard art-gallery problem [10]. The camera placement problems we consider are even more complex for many reasons including that they concern covering a 3D space rather than a 2D one. Because the underlying problem is NP-hard, efficient camera placement optimization can only be heuristic in the general case. Ram et al. [11] provide a camera placement optimization algorithm aimed at surveillance, but consider only the 2D problem. Hörster and Lienhart [12] provide optimization algorithms for four notions of coverage. While they only explicitly discuss the 2D case, their claim that their methods extend to 3D seems reasonable.

Camera calibration involves determining the intrinsic parameters of a camera, such as its focal length, principal point, and lens distortion, and its extrinsic parameters, indicating how it is placed in the world. Intrinsic parameters are most commonly determined using a calibration object with a known pattern (e.g., a checkerboard) that is placed in many positions in front of a camera to calculate the calibration parameters [13]. The use of corresponding sets of image points and world points for extrinsic camera calibration is also well known. For example, OpenCV includes the function CVFINDEXTRINSICCAMERAPARAMS2 that returns extrinsic calibration parameters upon input of intrinsic parameters and corresponding sets of image and world points [14]. (However, at least in the summer of 2006, we found it unreliable when the world points were not roughly planar.)

Graphical interfaces for establishing correspondence points are less common. Merritt [15] provides such an interface, but differs from ours in several ways. Our system suggests possible correspondences and uses only a single window with parts of the model superimposed on the image. Merrit's system requires separate windows onto the model and image and does not suggest possible correspondences. Our work may be related to that of Freeman et al. [16], but they give little detail of their interface.

3. SYSTEM ARCHITECTURE AND INPUT

This section describes the DOTS system architecture and the representation and entry of geometric input. The implementation of DOTS is distributed among various computers. It includes a MySQL database for storing configuration and analysis data, a Network Video Recorder (NVR) implemented in Java and running on Linux, and C++ based processes for handling single camera tracking and cross camera fusion. The analysis processes are distributed across a set of Windows XP servers by a configuration script which assigns each successive analysis process to the least loaded server. In our current configuration with 20 cameras, 8 analysis servers are used.

Although the primary 'end user' viewing tool is the Java-based viewer shown in Figure 1, the DOTS database schema and NVR HTTP interface serve as an API to DOTS, making it easy to integrate other tools or web interfaces. Currently the system includes additional time line, floor plan, and viewer tools written in Python and C++, such as a 3D Viewer.

DOTS implements single camera tracking using a foreground segmentation algorithm described in [4]. The algorithm uses a standard Gaussian Mixture background model to determine candidate foreground pixels, and then uses normalized cross correlation between the neighborhoods of candidates pixels in the input image and the background model for a final classification. Only if the normalized cross-correlation is less than a threshold is the candidate pixel classified as a

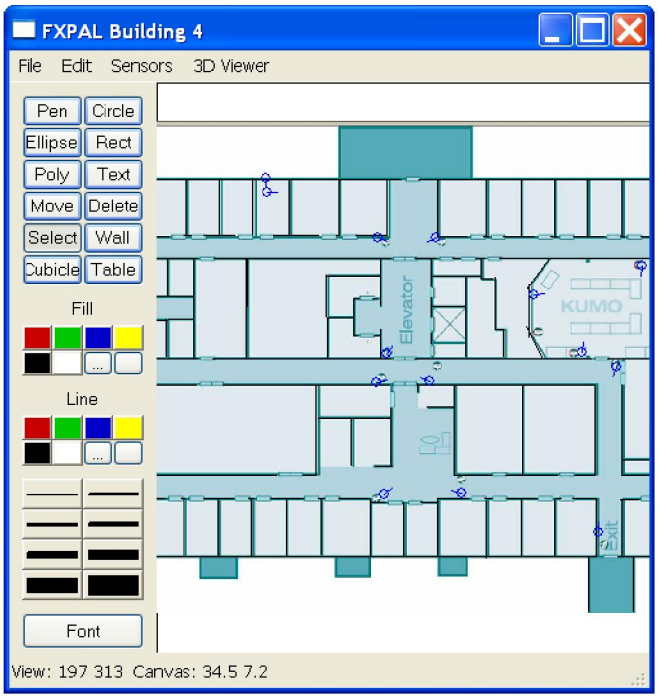


Fig. 2. Our tool for building a model from a floor plan

foreground pixel. We found this algorithm to be effective in suppressing segmentation errors due to shadows or changes in lighting or camera white balance and gain settings. The foreground segmentation is used both as the first step of the tracking algorithm, and to provide foreground segments shown in the 3D viewer.

The minimum geometric input to our DOTS system is a floor plan. Additional information, such as cubicle wall heights, is crucial in some settings and for some tasks. Other sorts of information, such as places of special interest or private places that should either not be visible in any camera view or only at low resolution, can be incorporated into DOTS.

Models are produced by a tool (Figure 2) that allows tracing over a floor plan image to define walls. Alternatively models could be imported from CAD or architectural files. Users may draw lines to define walls, cubicle dividers, tables, etc., or import predefined models such as chairs, desks, etc. Textures can be specified for the different surfaces, or simple default textures can be used. We are currently adding the capability to allow perspective corrected texture mapping from cameras that have been set up and calibrated, as is done in the video flashlight system [5, 17]. The floor may be texture-mapped with the building floor plan, or with default floor textures for carpet, tiles, etc. Estimates of the position and orientation of the cameras are also entered manually and then corrected using the tools described in Section 5.3 below.

The determined geometry is saved into a MySQL database and is also saved into VRML files. Our 3D visualization tools are built using the Coin3D implementation of Open Inventor, which can read VRML files as well as other file types [18]. The resulting VRML files can be edited if desired for more control over effects like lighting.

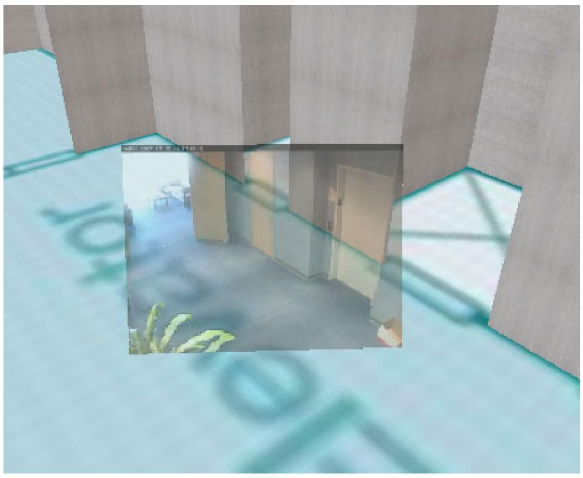


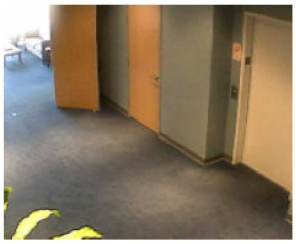
Fig. 3. Viewing the video from a camera when in the 3D model

4. GEOMETRIC UNDERSTANDING OF CAMERA VIEWS

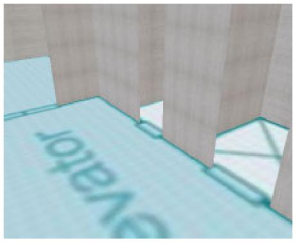
This section describes DOTS tools that aid interpretation of the spatial content of the camera views presented to surveillance workers. Video feeds from all cameras are placed within the 3D model to help users understand what is visible through each camera (Figure 3). The rendered video images are given the same orientation as the cameras. When viewed from behind, i.e. in the same direction as the camera is pointing, the images have the same left-right sense as the video, but when viewed from in front, the images are left-right reversed. For each camera, the interface supports quick access to two views: (1) the view from the camera and (2) a view above and behind each camera. The second view helps users orient themselves in space with respect to the camera. This interface allows the user to choose whether to see the view of the model from the camera position, the video feed from the camera, or a blend of the two (Figure 4). When the view is changed from one camera to another, a smooth interpolated path of the virtual camera helps the users "feel" the geometric relation between the two cameras.

The 3D model supports the DOTS surveillance system in a variety of ways, from providing alternative visualizations to estimating a person's position. Using 3D geometric information together with information from a tracker, we estimate a person's location in the space. Given an unoccluded view of a person (Figure 6), a reasonable assumption is that the bottom of a bounding box corresponds to the person's feet, so the person's position can be estimated by determining the in-

(a) Blend



(b) Video



(c) Model

Fig. 4. Blending of video and model views.

tersection of the optical ray from that point in the camera to the floor plane. More difficult, however, are situations when a person is semi-occluded, say, by cubicle walls. In those cases we find any walls whose top would map to within a few pixels of the bottom of the bounding box of a tracked person to find walls that may be occluding the person. We also find any surfaces that are occluded by the person. Figure 5 shows the bounding box for a tracked person together with the nearest walls that occlude the person and are occluded by the person. From this information, we can estimate the location of the tracked person within a cubicle area. Generally we take the projection onto the floor of the center point of the line segment obtained by intersecting the ray corresponding to the bottom center point of the bounding box and the area discovered. If additional information is known, for instance the person's height, a more accurate estimate can be obtained.

Segmented foreground regions of tracked people are displayed in the 3D model of the surveillance area; the segments are shown as "billboards" facing the virtual camera, placed at the tracked position of the person (Figure 6). Arbitrary viewpoints are supported. Two modes provide particularly useful mobile views. One mode places the virtual view at the position of a tracked person to help a surveillance user understand what the tracked person can see as they move. The other mode automatically chooses the best camera view of a tracked person. As the person moves around and the best camera view changes, the virtual view smoothly transitions to the next camera. Choosing a virtual view above and behind the camera helps users understand the spatial context of the camera. For example, Figure 6 shows a "billboard" of

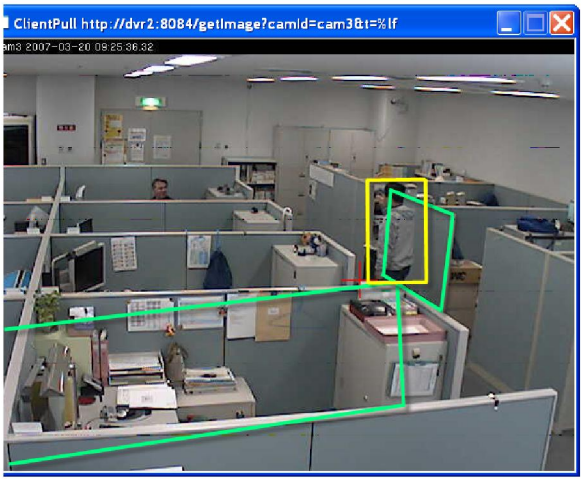


Fig. 5. Output from tracker shows a bounding box from which cubicle walls immediately in front and behind the person are calculated (shown). Location on floor is estimated from these.

a tracked person in the model from a virtual position slightly above and behind the camera from which the image was taken and the view in the model corresponding to what the tracked person sees.

5. GEOMETRIC SUPPORT FOR CAMERA SYSTEM DEPLOYMENT AND ENHANCEMENT

In addition to being a good way to view video, we found the 3D viewer useful for administration of the DOTS surveillance system; this section describes how tools in DOTS help in selecting good positions for camera placement and in calibrating the cameras.

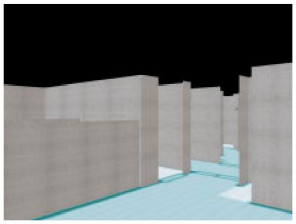
We developed several 2D tools to support calibration and placement of cameras. A map indicates camera locations, orientations, and views. All camera locations are shown on a map (Figure 8). Arrows indicate the pan directions of the cameras (Figure 7). The tool shown in Figure 7 provides direct manipulation of a camera's placement and orientation on the map; dragging the dot enables repositioning of the camera's 2D position on the map, and dragging the arrow changes the pan. Clicking on the camera brings up a video display showing a live view from a camera. A cone displays the floor area visible from a camera (Figure 7). The cone takes walls and the camera orientation into consideration; the camera tilt in particular influences the minimum and maximum distance from which the floor is visible. The cone fades with distance to indicate the quality of the picture (Figure 8). The cone fades more quickly for wide-angle cameras.

5.1. Enhanced floor plan for camera placement decisions

The goodness of a camera placement depends on the task at hand. In realistic situations a number of competing desires are

(a)

(b)



(c)

Fig. 6. Views related to a tracked person: (a) Camera view of a tracked person, (b) Model view above and behind corresponding virtual camera, and (c) The view of the model from the tracked person's point of view.

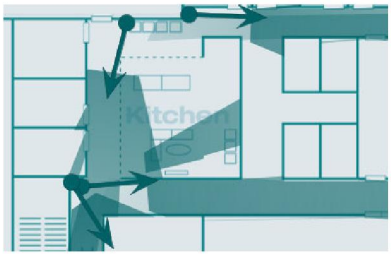


Fig. 7. Representation of cameras and their orientations on the floor plan interface. Manipulation of camera position and orientation on the map can be done by dragging the dot and arrow respectively.

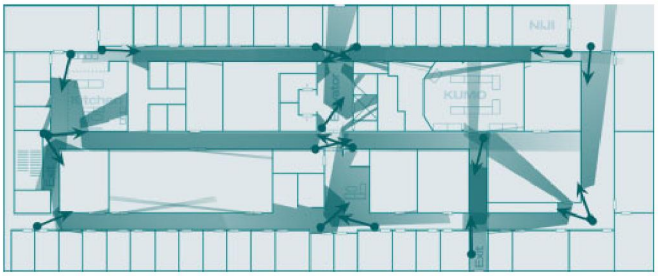


Fig. 8. Floor coverage

in play and vary in importance between locations. Users often have strong intuition for what makes good camera placement which they find difficult to quantify. Finding good quality metrics remains an active area of research; for example, the main thrust of Chen and Davis [19] is a quality metric for a 3D motion capture application, and the recent paper by Ram et al. [11] proposes a novel performance metric suitable for certain surveillance tasks. For these reasons, many tasks are not adequately supported by automatic camera placement methods. On the other hand exploring camera placement options by physically mounting cameras is too burdensome to be pursued realistically in most cases.

Our tools support users in efficiently exploring camera placement options in a virtual model by providing visualizations for certain measures of goodness, including various of notions of coverage, while at the same time providing users with ways of experiencing a camera placement so that they can make use of their intuitions. Coverage is often a top concern but different notions of coverage may be more or less useful in different situations: how much of the space is visible to head height; is close to a camera; is visible to multiple cameras so that triangulation is possible? In cases where only sparse coverage is possible, how should cameras be placed to obtain good estimates of a tracked person's off-camera position? Users may have fine grained notions of the relative importance of types of coverage in different locations in the space, and which trade-offs are worthwhile. Other consideration need to be taken into account such as how easily a camera can be mounted in a location, how conspicuous it is, and, for security applications, how easily it could be tampered with.

The direct manipulation of camera positions on the map combined with the visualization of visible areas provide a convenient way for planning camera coverage. Users can move and rotate cameras on the map until they are satisfied with the coverage. Instead of fading cones, users can also select solid cones to better judge what is visible at all. Furthermore, users can specify a distance at which the cones will be truncated in order to see how much of the space is covered by nearby cameras. Users can also adjust the height at which the coverage is judged (Figure 9). The selected distance is adjusted appropriately for wide-angle cameras so that a person standing at the cut-off distance would appear the same size in all cameras. Finally, we also offer a view of all areas that are visible by at least two cameras (Figure 10).

5.2. 3D environment for placing cameras

Even before any cameras have been installed, virtual camera views can be obtained from the 3D model. Figure 11 shows the view from the virtual camera at the position shown on the map. These views give users an impression of what they would see if they placed cameras at particular places. The virtual camera view can be controlled by dragging an indicator on the floor plan view or manipulating the virtual camera view window with a mouse or joystick.

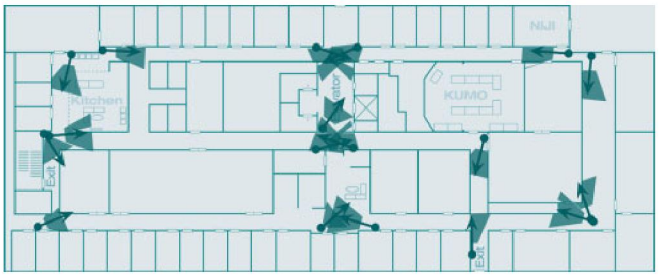


Fig. 9. Areas in which head height (1.5m) is within 5 meters of some camera.

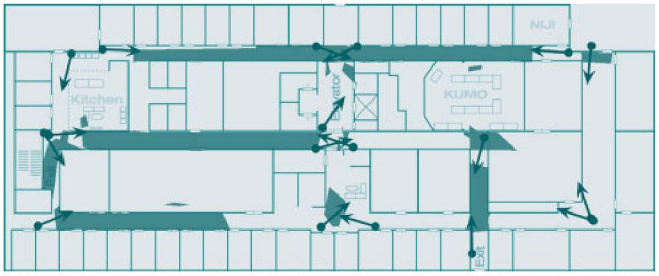


Fig. 10. Floor covered by at least two cameras.

Similar to the 2D cones on the floor plan described above, the 3D model can also visualize what cameras can see. The parts of the model visible to a camera or set of camera can be illuminated (Figure 12). This provides a different perspective that is helpful in evaluating camera placement and orientation, particularly when there are downward pointing cameras, or complex occlusions. As a camera is moved around, the 'illuminated area' visible to that camera is updated in real time, using a shadow engine which takes advantage of the Graphics Processing Unit. We used a shadow engine for Open Inventor, provide by [20] and based on the 'shadow volume' algorithm developed by [21]. This visualization allows users to understand which areas visible from one camera are or are not visible in other cameras.

5.3. Geometric support for camera calibration

Two different methods are provided for finer-grained calibration. Both assume a reasonable initial estimate of camera position and orientation, which can be entered numerically or by the direct manipulation tool described above. We are most concerned with extrinsic calibration since intrinsic camera parameters can be determined prior to set up using standard techniques. The first method provides visualization for hand adjustment of the camera parameters. The second method provides an easy way to establish correspondence points between a camera image and the 3D model that can be used to automatically determine the calibration parameters. We have found both approaches useful.

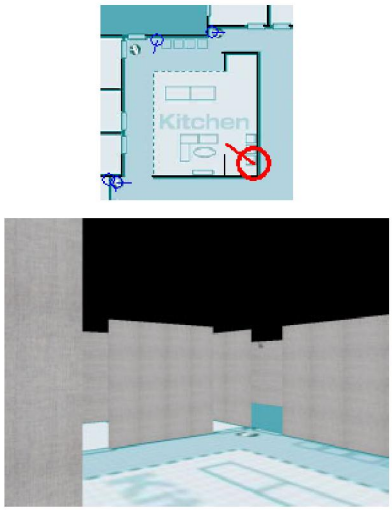


Fig. 11. View from a virtual camera placed at the position shown on map.

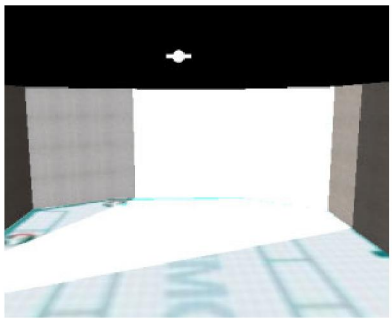


Fig. 12. View showing camera coverage of area in 3D

Wall outlines from the model can be superimposed on the video view, using the current estimate of the camera parameters, to assess the camera calibration. Sliders allow the manipulation of the camera location, orientation, and view angle (Figure 13). Wall outlines show whether the real camera view matches the modeled view. If the camera distortion is known, it can be used to show curved wall outlines. Otherwise, we use a pinhole model and show straight wall outlines. All camera parameters can be adjusted via sliders. Additionally, the camera pan and tilt can be adjusted directly by dragging the mouse across the image. The results of slider changes are displayed immediately on the map and the wall outlines. Using the sliders, one can adjust the camera parameters until the walls match the live video image. However, this adjustment can be quite challenging and time consuming.

Hand adjustment can be quite difficult. One source of confusion is that several parameters have similar effects but are subtly different; for example, adjusting the field of view and moving the floor position along the view axis. Also 3D orientations can be quite confusing; adjusting cameras with non-trivial roll was particularly challenging. In general, ad-

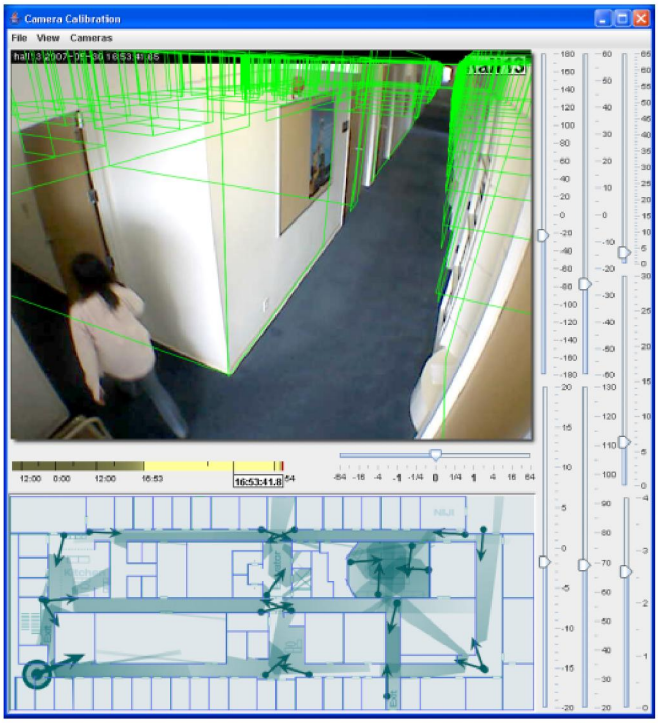


Fig. 13. Calibration with sliders

justing camera parameters to match the video image can be frustrating. Wide-angle cameras prove particularly challenging because they have more distortion. It is difficult to decide which parameter should be changed to improve the calibration. However, having wall outlines instead of just correspondence points makes the task easier. While adjusting the cameras, we found that the measured camera positions were less accurate than we believed so that the horizontal camera position had to be adjusted as well.

An alternative method for fine grained camera calibration is provided by our correspondence point tool. Again wall outlines for the model are superimposed on the video view, but in this case we only superimpose one rectangle at a time. If the user clicks near a corner of the rectangle, a line segment extending from that corner is drawn. The user may then position the end of the line segment over the position in the image that corresponds to the corner (Figure 14). These correspondences can then be fed to a program that minimizes the least squares distance over camera parameterizations between the image points and computed estimates for the images of the corresponding 3D points given a parameterization. A standard hill climbing algorithm (in our case, the fmin function of SciPy [22]) gave excellent results. We used about a dozen points and minimized over seven parameters: the 3D position, orientation, and focal length. Excellent results could often be achieved with fewer points. Some camera views have few corners that can be used to establish correspondence points. In such cases, only hand adjustment using sliders can be used.

Fig. 14. Calibration with correspondence points tool

6. CONCLUSIONS AND FUTURE WORK

We have described our geometric tools and how they have been used as part of our DOTS surveillance system. We used these tools both during deployment and during surveillance itself.

Our tools assist camera placement decisions, both for initial deployment and as cameras are added; and they provide calibration support using the 3D model. Instead of providing automatic planning tools for camera placement, the tools visualize the consequences of placements to the user in a direct manipulation tool that enables the user to quickly explore alternatives. Direct user manipulation of the camera placement/calibration is useful because automatic planning may not deal with constraints of surveillance tasks very well. Once cameras have been installed, our calibration tools help users determine accurate camera positions, orientations and adjust focal lengths. The tools overlay wall outlines with the 3D model on the video display. In one tool, these outlines move as the user adjusts the camera parameters. A second tool allows users to establish correspondences between points in the video and the projection of the 3D model, and uses the correspondences to optimize the camera parameters. These tools make it feasible to install our DOTS surveillance system at different sites easily, as opposed to the traditional time consuming trial-and-error manner.

Our DOTS system also uses the 3D model to provide context to a surveillance worker trying to make sense of many camera views. The video feed from a camera can be shown within the 3D model at a place appropriate to the camera position, enabling the worker to place the camera feed within the 3D context. The three-dimensional geometric information allows estimates of a person's location even when that person is semi-occluded, e.g., by cubicle walls. Segmented regions of tracked people are displayed at the estimated location in the model; and modes are provided to allow a surveillance worker

to follow a tracked person through the model, or to request the view from the tracked person's vantage point.

While we have used our tools only for static cameras that can be hand rotated, many of these tools can be used and further developed, to work with cameras that can be automatically rotated, moved along a track, or mounted on a roving robot. We will also experiment with the incorporation of information from other sensors, like sensors that can tell whether a door is open or closed. We also intend to incorporate tools that support more automated camera calibration and placement decisions while retaining the ability of the user to directly manipulate and visualize the outcomes of various decisions. Finally we intend to further enhance our DOTS tracking system with more sophisticated use of geometric information.

7. REFERENCES

[1] A. Girgensohn, D. Kimber, J. Vaughan, T. Yang, F. Shipman, T. Turner, E. Rieffel, L. Wilcox, F. Chen, and T. Dunnigan, "DOTS: Support for effective video surveillance," *Proceedings of ACM Multimedia*, 2007.

[2] A. Girgensohn, F. Shipman, and T. T. L. Wilcox, "Effects of presenting geographical context on tracking activity between cameras," *CHI 2007*, 2007.

[3] D. Kimber, T. Dunnigan, A. Girgensohn, F. Shipman, T. Turner, and T. Yang, "Trailblazing: video playback control by direct object manipulation," *ICME 2007*, 2007.

[4] T. Yang, F. Chen, D. Kimber, and J. Vaughan, "Robust people detection and tracking in a multi-camera indoor visual surveillance system," *ICME 2007*, 2007.

[5] "VistaScape Security System," http://www.vistascape.com/.

[6] "L3 Communications," http://www.l3praetorian.com/.

[7] I. O. Sebe, J. Hu, S. You, and U. Neumann, "3d video surveillance with augmented virtual environments," in *IWVS '03: First ACM SIGMM international workshop on Video surveillance*, New York, NY, USA, 2003, pp. 107–112, ACM Press.

[8] A. State, G. Welch, and A. Ilie, "An interactive camera placement and visibility simulator for image-based VR applications," in *Stereoscopic Displays and Virtual Reality Systems XIII. Proceedings of the SPIE.*, A. J. Woods, N. A. Dodgson, J. O. Merritt, M. T. Bolas, and I. E. McDowall, Eds., Feb. 2006, vol. 6055, pp. 640–651.

[9] J. Williams and W.-S. Lee, "Interactive virtual simulation for multiple camera placement," in *Haptic Audio Visual Environments and their Applications*, 2006, pp. 124–129.

[10] J. O'Rourke, *Art gallery theorems and algorithms*, Oxford University Press, 1987.

[11] S. Ram, K. R. Ramakrishnan, P. K. Atrey, V. K. Singh, and M. S. Kankanhalli, "A design methodology for selection and placement of sensors in multimedia surveillance systems," in *VSSN '06: Proceedings of the 4th ACM international workshop on Video surveillance and sensor networks*, New York, NY, USA, October 2006, pp. 121–130, ACM Press.

[12] E. Hörster and R. Lienhart, "On the optimal placement of multiple visual sensors," in *VSSN '06: Proceedings of the 4th ACM international workshop on Video surveillance and sensor networks*, New York, NY, USA, October 2006, pp. 111–120, ACM Press.

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

[14] "OpenCV," http://opencvlibrary.sourceforge.net/CvReference.

[15] J. Merritt, "Camera calibration for blender," http://jmerritt.warpax.com/pytsai.

[16] R. Freeman, A. Steed, and B. Zhou, "Rapid scene modelling, registration and specification for mixed reality systems," in *VRST '05: Proceedings of the ACM symposium on Virtual reality software and technology*, 2005, pp. 147–150.

[17] H. S. Sawhney, A. Arpa, R. Kumar, S. Samarasekera, M. Aggarwal, S. Hsu, D. Nister, and K. Hanna, "Video flashlights: real time rendering of multiple videos for immersive model visualization," in *EGRW '02: Proceedings of the 13th Eurographics workshop on Rendering*, 2002, pp. 157–168.

[18] "Coin3D," http://coin3d.org/.

[19] X. Chen and J. Davis, "Camera placement considering occlusion for robust motion capture," Stanford Computer Science Technical Report, 2000.

[20] T. Burian, "Realtime shadow algorithms," M.S. thesis, Brno University of Technology, 2006.

[21] F. C. Crow, "Shadow algorithm for computer graphics," in *Proceedings of SIGGRAPH*, 1977, pp. 242–248.

[22] "SciPy," http://www.scipy.org.