3D Surveillance – A Distributed Network of Smart Cameras for Real-Time Tracking and its Visualization in 3D

Sven Fleck, Florian Busch, Peter Biber, Wolfgang Straßer
WSI/GRIS, University of Tübingen
Sand 14, 72076 Tübingen, Germany
{fleck,busch,biber,strasser}@gris.uni-tuebingen.de
http://www.gris.uni-tuebingen.de/staff/Sven_Fleck.html

Abstract

The demand for surveillance systems has increased extremely over recent times. We present a system consisting of a distributed network of cameras that allows for tracking and handover of multiple persons in real time. The inter-camera tracking results are embedded as live textures in an integrated 3D world model which is available ubiquitously and can be viewed from arbitrary perspectives independent of the persons' movements. We mainly concentrate on our implementation of embedded camera nodes in the form of smart cameras and discuss the benefits of such a distributed surveillance network compared to a host centralized approach. We also briefly describe our way of hassle free 3D model acquisition to cover the complete system from setup to operation and finally show some results of both an indoor and an outdoor system in operation.

1. Introduction

In typical surveillance systems today, the raw live video stream of a huge number of cameras is displayed on a set of monitors, so the security personnel can respond to situations accordingly. For example, in a typical Las Vegas casino, approximately 1,700 cameras are installed [1]. If you want to track a suspect on his way, you have to manually follow him within a certain camera. Additionally, when he leaves one camera's view, you have to switch to an appropriate camera manually and put yourself in the new point of view to keep up tracking. A more intuitive 3D visualization where the person's path is integrated from a distributed network of smart cameras in one consistent world model, independent of all cameras, is not yet available.

Imagine a distributed, inter-sensor surveillance system that reflects the world and its events in an integrated 3D world model which can be visualized ubiquitously within the network, independent of camera views. This dream includes a hassle free and automated method for acquiring a 3D model of the environment of interest, an easy plug 'n' play style of adding new smart camera nodes to the network, the distributed tracking and person handover itself and the integration of all cameras' tracking results in one consistent model.

In this paper we present a surveillance system to come a little bit closer to this dream. The system consists of a virtually arbitrary number of camera nodes, a server node and a visualization node. Each camera node is realized by a smart camera or a combination of one or more cameras and a PC. The former is of course the preferred implementation that offers various benefits and will thus be described later in detail, the latter extends the use of our system and ensures an easier migration from existing camera systems. A heterogenous network of both kinds of nodes is also possible. To cover the whole system, our contribution does not stop from presenting an easy method for 3D model acquisition of both indoor and outdoor scenes as content for the visualization node by the use of our mobile platform – the Wägele. Results of both indoor and outdoor setups are presented in section 3 before we conclude this paper.

1.1. Related Work

1.1.1 Surveillance Systems

The IEEE Signal Processing issue on surveillance [2] surveys the current status of surveillance systems, e.g., Foresti et al. present "active video based surveillance systems", Hampur et al. describe their multiscale tracking system. On CVPR05, Terry Boult gave an excellent tutorial of surveillance methods [3]. Siebel et. al especially deal with the problem of multi camera tracking and person handover within the ADVISOR surveillance system [4]. Trivedi et. al presented a distributed video array for situation awareness [5] that also gives a great overview about the current state of the art of surveillance systems. Yang et. al [6] describe a camera network for real time people counting in crowds.

The Sarnoff group presented an interesting system called "video flashlight" [7] where the output of traditional cameras are used as live texture mapped onto the ground/walls of a 3D model.

However, the idea of a surveillance system consisting of a distributed network of smart cameras and a live visualization of tracking results embedded in a 3D model has not been covered yet.

1.1.2 Smart Cameras

There exists a variety of smart camera architectures designed in academia [8, 9] and industry. What all smart cameras share is the combination of a sensor, an embedded processing unit and a connection, which is nowadays often a network unit. The processing means can be roughly classified in DSPs, general purpose processors, FPGAs and a combination thereof. The idea of having Linux running embedded on the smart camera gets more and more common (Matrix Vision, Basler).

From the other side, the surveillance sector, IP based cameras are emerging where the primary goal is to transmit live video streams to the network by self contained camera units with (often wireless) Ethernet connection and embedded processing that deals with the image acquisition, compression (MJPEG or MPEG4), a webserver and the TCP/IP stack and offer a plug 'n play solution. Further processing is typically restricted to, e.g., user definable motion detection. All the underlying computation resources are normally hidden from the user.

The border between the two classes gets more and more fuzzy, as the machine vision originated smart cameras get (often even GigaBit) Ethernet connection and on the other hand the IP cameras get more computing power and user accessability to the processing resources. For example the ETRAX100LX processors of the Axis IP cameras are fully accessible and also run Linux.

Our goal is to use a smart camera based surveillance network. Only very few surveillance systems actually use smart cameras yet.

1.1.3 Tracking – Particle Filter

Tracking as one key component of our system is based on particle filters. Particle filters have become a major way of tracking objects [10, 11]. The IEEE special issue [12] gives a good overview of the state of the art . Utilized visual cues include shape [11] and color [13, 14, 15] or a fusion of cues.

2. 3D Surveillance System Architecture

2.1. Architecture Overview

The top level architecture of our distributed surveillance and visualization system is given in Fig. 1. It consists of multiple, networking enabled camera nodes, a server node and a 3D visualization node. Each camera node is implemented either by a smart camera or by a combination of one or multiple non-smart cameras in combination with a PC to allow for maximum flexibility. In the following, all components are described on top level, before each of them is detailed in the following sections.

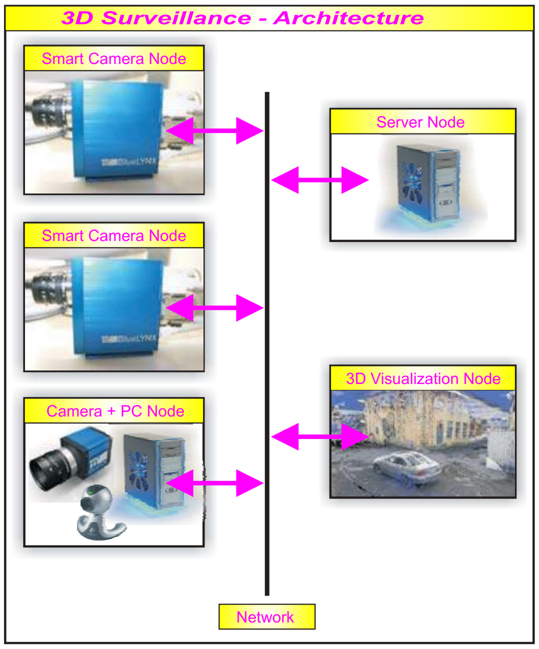


Figure 1. 3D Surveillance System Architecture

2.1.1 Smart Camera Nodes

Our work is based on mvBlueLYNX smart cameras from from Matrix Vision [16] like shown in Fig. 2. The 420CX we use contain single CCD sensors with VGA resolution

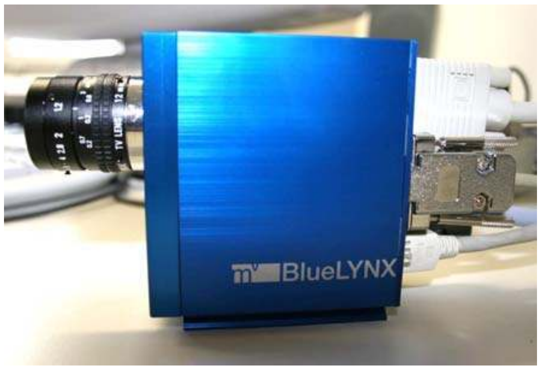


Figure 2. A smart camera node of our system

(progressive scan, 12 MHz pixel clock) and attached Bayer color mosaics. Xilinx Spartan-IIE FPGAs (XC2S400E) are used for low-level processing. 200 MHz Motorola MPC 8241 PowerPC processors with MMU & FPU running embedded Linux are utilized for the main computations. They further comprise 32 MB SDRAM (64 Bit, 100 MHz), 32 MB NOR-FLASH (4 MB Linux system files, approx. 40 MB compressed user filesystem) and 4 MB NAND-FLASH (bootloader, kernel, safeboot system, system configuration parameters). The smart cameras communicate via 100 MBit/s Ethernet connections, which are used both for field upgradeability and parameterization of the system and for transmission of the tracking results during runtime. See [17, 18] for further details.

2.1.2 PC based Camera Nodes

Besides the preferred realization as smart camera, our system also allows for using standard cameras in combination with a PC to form a camera node. Even multiple standard cameras can be attached and computed by a single PC. We implemented a DirectShow interface for the PC based nodes, so any DirectShow device can be used: industrial cameras (like the Matrix Vision mvBlueFox), webcams and also analog cameras attached to a capture card which enables a more hassle free migration from deprecated installations. We demonstrate the capability for heterogenous systems in terms of camera nodes in our results section 3.

2.1.3 Server Node

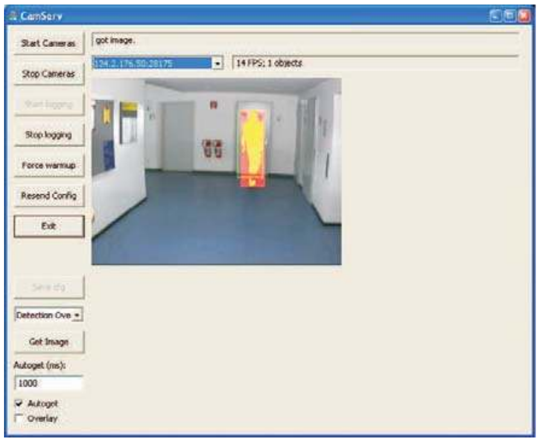
The server node acts as server for all the camera nodes and concurrently as client for the visualization node. It manages configuration and initialization of all camera nodes, collects the resulting tracking data and takes care of person handover. Fig. 3 illustrates a screenshot of the server node.

2.1.4 Visualization Node

The visualization node acts as server, receiving position, size and texture of each object currently tracked by any camera from the server node. It embeds the ROI of each object as a sprite (using the live camera texture) in a rendered 3D point cloud of the environment (see Fig. 4). Both the visualization node and the server node can run together on a single PC.

2.2. Smart Camera Node in Detail

The smart camera tracking architecture as one key component of our system is illustrated in Fig. 5 and comprises the following components: a background modeling & auto init unit, multiple instances of a particle filter based tracking unit, $2D \rightarrow 3D$ conversion units and a network unit.



got image.

14 FPS; 1 objects

Figure 3. GUI of the Server Node. Note that the raw camera stream is only updated for the camera currently selected at an adjustable interval to save bandwidth.

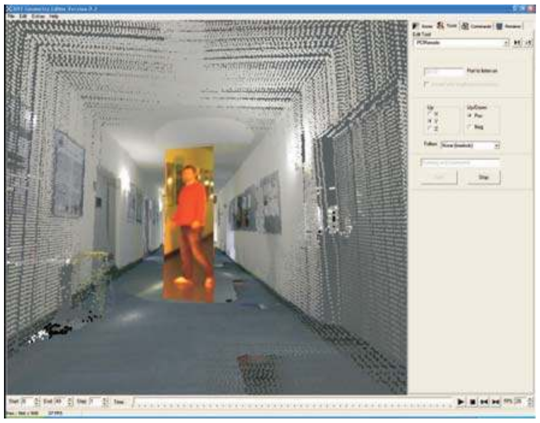


Figure 4. GUI of the XRT based Visualization Node where a person currently tracked is embedded as live texture at his estimated position.

2.2.1 Background Modeling – Segmentation – Auto Init

In contrast to our previous system [17, 18], we take advantage of the fact that each camera is mounted statically. This enables the use of a background model for segmentation of moving objects. As we target an embedded implementation, computation time is very crucial so we developed a high speed background unit as illustrated in Fig. 6. For every pixel, the noise process (4) is estimated based on previous observations and a change threshold is determined via a linear function. The actual image (1) is compared to an old image (2). If the difference (3) is higher than this threshold, the counter (5) is reset to zero, otherwise it is increased. Once the counter reaches a certain level, it triggers the updating of

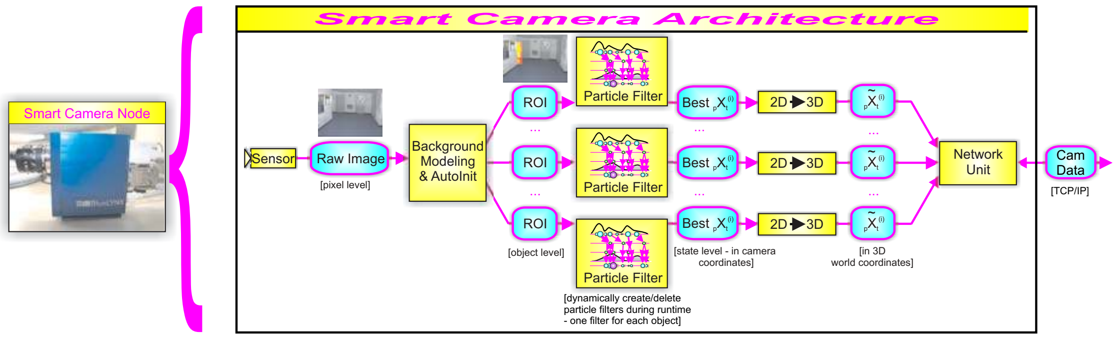

Smart Camera Architecture
Smart Camera Node
Sensor
Raw Image
[pixel level]
Background Modeling & AutoInit
ROI
[object level]
Particle Filter
[dynamically create/delete particle filters during runtime - one filter for each object]
Best
[state level - in camera coordinates]
2D▶3D
[in 3D world coordinates]
Network Unit
Cam Data
[TCP/IP]

Figure 5. Smart Camera Node's Architecture

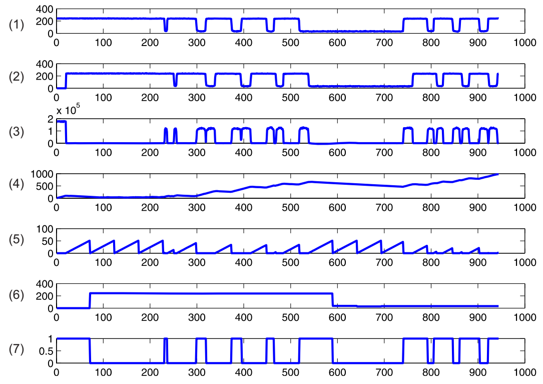


Figure 6. High speed background modeling unit in action. Per pixel: **(1)** Raw pixel signal from camera sensor. **(2)** 10 frames old raw signal. **(3)** Difference between (1) & (2). **(4)** Estimated noise process based on running average filtered first deviation of raw pixel data. **(5)** Confidence counter: Increased if pixel is consistent with background within a certain tolerance, reset otherwise. **(6)** Background model. **(7)** Trigger event if motion is detected.

the background model (6) at this pixel. Additionally, it is reset back to zero for speed purposes (to circumvent adaption and thus additional memory operations at every frame). For illustration purposes, the time it takes to update the background model is set to 50 frames in Fig. 6 (see first rising edge in (6)). The background is updated every time the confidence counter (5) reaches 50. The fluctuations of (1) up until $t = 540$ are not long enough to update the background model and are hence marked as moving pixels in (7). At $t = 590$ the difference (3) kept low for 50 frames sustained, so the background model is updated (in (6)) and the pixel is no longer marked as moving (7). The fluctuations towards the end are then classified as moving pixels. Single pixels are then eliminated by a 4-neighborhood erosion. From the resulting mask of movements, areas are constructed via a region growing algorithm, finally leading to quadratic regions of interest containing the movement. This unit alone is capable of running at about 19 fps on the mvBlueLynx 420CX smart camera using QVGA resolution derived from the raw Bayer mosaiced VGA sensor data. The background modeling unit thus handles the transformation from raw pixel level to object level. For each detected object yet untracked, a new particle filter is dynamically created during runtime whereas the current appearance is used as target appearance. This enables the tracking of multiple objects, which is described next. The according particle filter is destroyed if the object is not available any more over a certain time.

2.2.2 Multi Object Tracking – Color based Particle Filters

For each person/object p a particle filter engine is instantiated. Particle filters have become an attractive way of tracking, as they are capable of handling multiple hypotheses (multimodal pdfs) and nonlinear systems. Fig. 7 illustrates the operation of a particle filter with 8 particles. Our engine is based on color histograms in HSV space, see [17, 18] for details. The state ${}_pX_t^{(i)}$ of sample i at time t comprises its position, size and velocity. It is also capable of adaptation of appearance during tracking. In contrast to [17] the confidence for adaptation comes from the background unit. On the mvBlueLynx smart camera, we achieve about 17 fps for the whole tracking pipeline when one object is tracked, 15 fps for two objects. Besides the number of objects, i.e., the number of particle filter instances, the frame rate is also affected by the target size on pixel level. We use a subsampling approach to attenuate this effect.

The tracked approximated pdf $p({}_pX_t|Z_t)$ is then reduced: only the maximum likelihood sample ${}_p\tilde{X}_t^{(i)}$, that represents the most probable position and scale for each object/person p in camera coordinates, is further processed in the $2D \rightarrow 3D$ conversion unit described next.

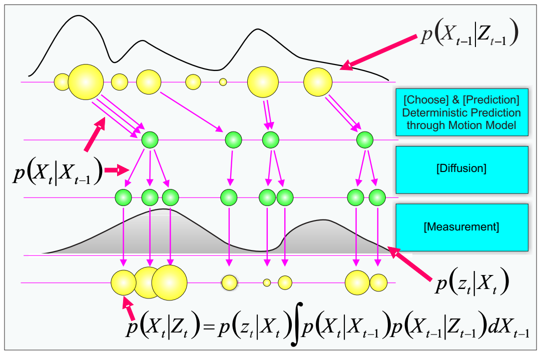

$p(X_{t-1}|Z_{t-1})$
[Choose] & [Prediction] Deterministic Prediction through Motion Model
$p(X_t|X_{t-1})$
[Diffusion]
[Measurement]
$p(z_t|X_t)$
$p(X_t|Z_t) = p(z_t|X_t)\int p(X_t|X_{t-1})p(X_{t-1}|Z_{t-1})dX_{t-1}$

Figure 7. Particle filter iteration loop. The size of each sample ${}_pX_t^{(i)}$ corresponds to its weight ${}_p\pi_t^{(i)}$.

2.2.3 $2D \rightarrow 3D$ Conversion Unit

3D Tracking is implemented by converting the 2D tracking results in image domain of the camera to a 3D world coordinate system with respect to the (potentially georeferenced) 3D model, which also enables global, inter-camera handling and handover of objects.

Since both external and internal camera parameters are known, we can convert 2D pixel coordinates into world coordinate view rays. The view rays of the lower left and right corner of the object are intersected with the fixed ground plane. The distance between them determines the width and the mean determines the position of the object. The height (e.g., of a person) is calculated by intersecting the view ray from the top center pixel with the plane perpendicular to the ground plane that goes through the two intersection points from the first step. If the object's region of interest is above the horizon, the detected position lies behind the camera and it will be ignored. The extracted data is then sent to the server along with the texture of the object.

2.3. Server Node in Detail

The server node is illustrated in Fig. 8. It consists of a camera protocol server, a camera GUI, a person handover unit and an XRT protocol client.

2.3.1 Camera Protocol Server

The camera server implements a binary protocol for communication with each camera node based on TCP/IP. It serves as sink for all camera nodes' tracking result streams, which consist of the actual tracking position and appearance (texture) of every target per camera node in world coordinates. This information is forwarded both to the person handover unit and to a log file that allows for debugging and playback of recorded data. Additionally, raw camera images can be acquired from any camera node for the camera GUI.

2.3.2 Camera GUI

The camera GUI visualizes all the results of any camera node: the segmented and tracked objects are overlayed over the raw sensor image. The update rate can be manually adjusted to save bandwidth. Additionally, the camera GUI supports easy calibration relative to the model by blending the rendered image of a virtual camera over the current live image as basis for optimal calibration, as illustrated in Fig. 9.

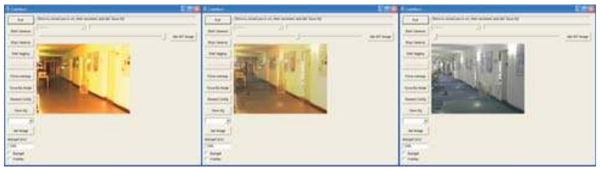


Figure 9. Camera GUI. Different blending levels are shown: **Left:** Real raw sensor image. **Right:** Rendered scene from same viewpoint.

2.3.3 Person Handover Unit

To achieve a seamless inter-camera tracking decoupled from each respective sensor node, the person handover unit merges objects tracked by different camera nodes if certain conditions are met. As results are already in world coordinates, they can be compared directly. After a new object has been detected by a camera node, its tracking position is compared to all other objects that are already being tracked. If an object at a similar position is found, it is considered the same object and statically linked to it using its global id.

2.3.4 XRT Protocol Client

The XRT client unit implements a TCP/IP based protocol to communicate the final objects tracked to the XRT visualization node.

2.4. Visualization Node in Detail

A practical 3D surveillance system also comprises an easy way of acquiring 3D models of the respective environment. Hence, we briefly present our 3D model acquisition system that provides the content for the visualization node which is described afterwards.

2.4.1 3D Model Acquisition for 3D Visualization

The basis for 3D model acquisition is our mobile platform which we call the Wägele[1] [19]. It allows for an easy ac-

[1] *Wägele* – Swabian for a little cart

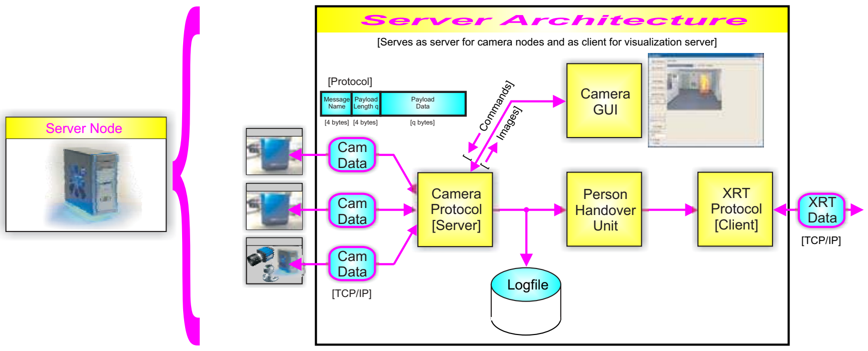


Figure 8. Server Architecture

quisition of indoor and outdoor scenes: 3D models are acquired just by moving the platform through the scene to be captured. Thereby, geometry is acquired continuously and color images are taken in regular intervals. Our platform (see Fig. 10) comprises an 8 MegaPixel omnidirectional camera (C1 in Fig. 10) in conjunction with three laser scanners (L1-L3 in Fig. 10) and an attitude heading sensor (A1 in Fig. 10). Two flows are implemented to yield 3D models: a computer vision flow based on graph cut stereo and a laser scanner based modeling flow. After a recording session the collected data is assembled to create a consistent 3D model in an automated offline processing step. First a 2D map of the scene is built and all scans of the localization scanner (and the attitude heading sensor) are matched to this map. This is accomplished by probabilistic scan matching using a generative model. After this step the position and orientation of the Wägele is known for each time step. This data is then fed into the graph cut stereo pipeline and the laser scanner pipeline. The stereo pipeline computes dense depth maps using pairs of panoramic images taken from different positions. In contrast to classical multi camera based scene capturing techniques we require only one camera and sample the environment, which is much more inexpensive and is not limited in the size of the environment. The laser flow projects the data from laser scanners L2 and L3 into space using the results of the localization step. L2 and L3 together provide a full 360° vertical slice of the environment. The camera C1, then, yields the texture for the 3D models. More details can be found in [19, 20].

2.4.2 3D Visualization Framework – XRT

The visualization node gets its data from the server node and renders the information (all objects currently tracked) embedded in the 3D model. It is based on the eXperimental Rendering Toolkit (XRT) developed by Michael Wand et.al at our institute which is a modular framework for real time point based rendering. The viewpoint can be chosen arbitrarily. Also a fly-by mode is available that moves the viewpoint with a tracked person/object. Objects are displayed as sprites using live textures. Resulting renderings are shown in the following section.

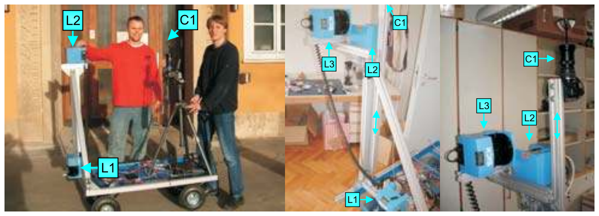


Figure 10. Two setups of our mobile platform. **Left:** Two Laser scanners L1, L2 and one omnidirectional camera. **Center & Right:** Three laser scanners L1, L2, L3 and omnidirectional camera closely mounted together.

3. Results

Two setups have been evaluated, an indoor setup in an office environment and an outdoor setup (see Fig. 11), more details and videos can be found on the project's website [21]. First, a 3D model of each environment has been acquired. Afterwards, the camera network has been set up and calibrated relative to the model. To circumvent strong reflections on the floor in the indoor setup, halogen lamps are used with similar directions as the camera viewpoints. Some indoor results are illustrated in Fig. 12. Fig. 13 shows results of the outdoor setup. Even under strong gusts where the trees were heavily moving, our per pixel noise process estimator enabled robust tracking by spatially adapting to the respective background movements.

4. Conclusion

Our approach of such a distributed network of smart cameras offers various benefits. In contrast to a host centralized approach, the possible number of cameras can eas-

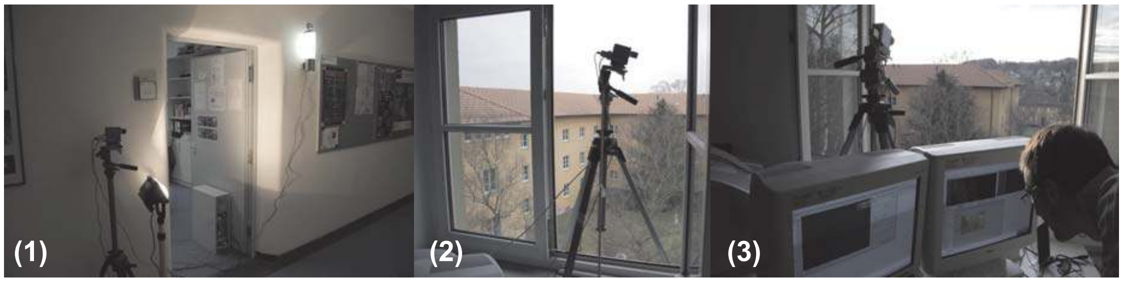


Figure 11. (**1**) Indoor Setup. Note the smart camera on the tripod, halogen illumination and a webcam on top of a light. (**2, 3**) Outdoor setup.

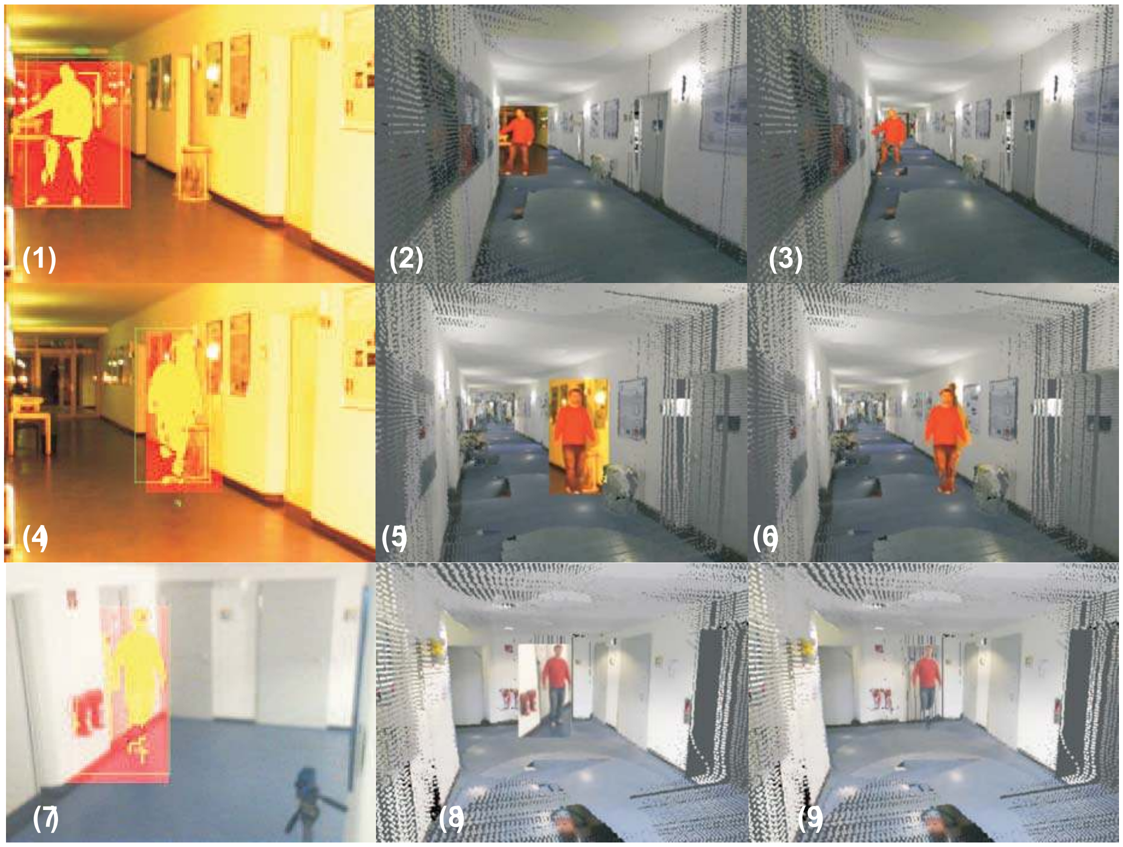


Figure 12. Indoor Setup. Renderings of the XRT Visualization Node. **Left column:** Output of the server node (Camera GUI): raw image of the camera node, overlayed with the target object on which a particle filter is running. **(1),(4):** Smart Camera, **(7):** webcam. **Center Column:** Rendering of embedded live texture in XRT visualization system. **Right Column:** Same as center, but with alpha map enabled: only segmented areas are overlayed for increased realism.

ily exceed hundreds. Neither the computation power of a host nor the physical cable length (e.g., like with Camera-Link) is a limiting factor. As the whole tracking is embedded inside each smart camera node, only very limited bandwidth is necessary which makes the use of Ethernet possible. Additionally, the possibility to combine standard cameras and PCs to form local camera nodes extends the use of PC based surveillance over larger areas where no smart cameras are available yet. Our 3D visualization of tracking results enables a more intuitive and inter-camera visualization of tracked persons. Research in this direction is also important as security personnel have to concentrate

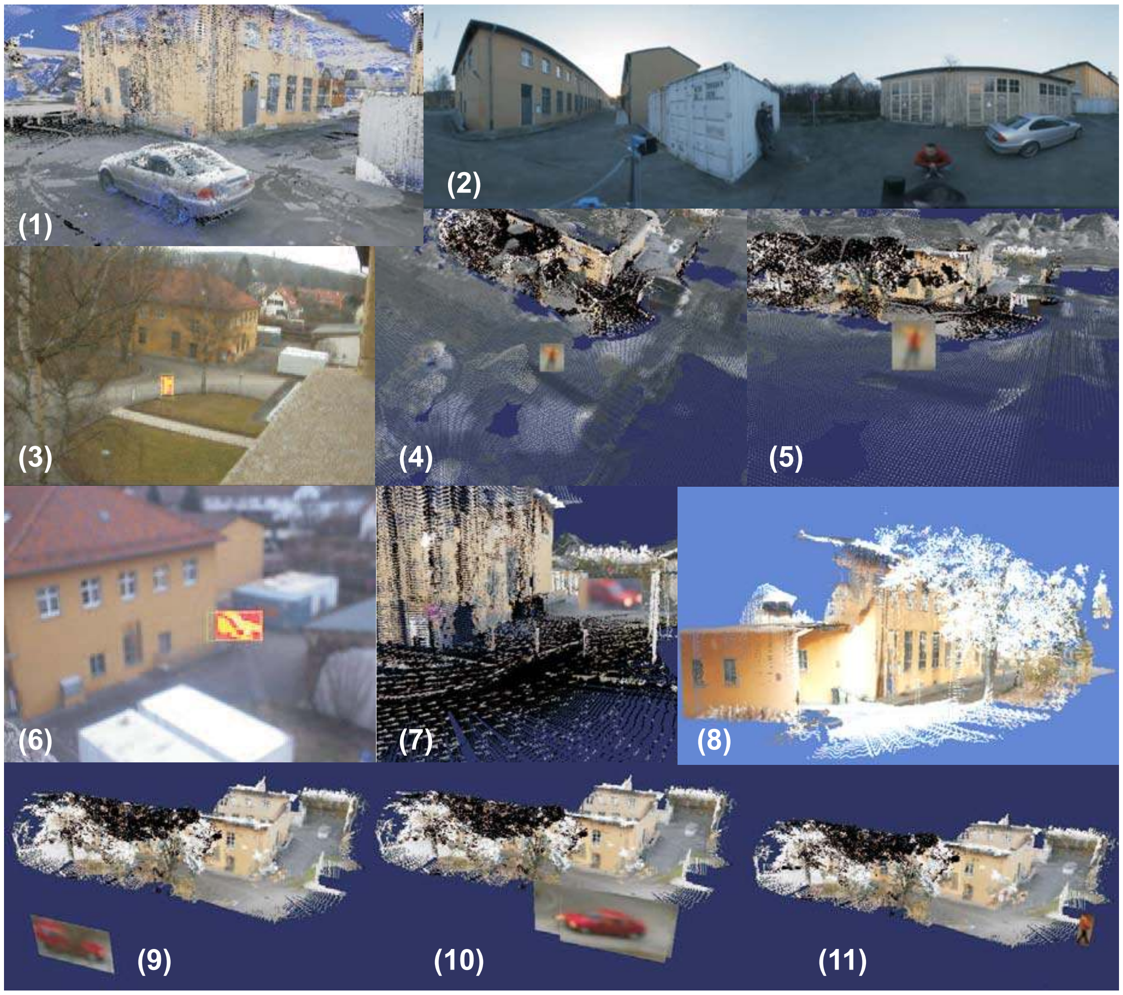


Figure 13. Outdoor setup **(1,8):** Renderings of the acquired model in XRT visualization system. **(2):** Dewarped example of an omnidirectional image of the model acquisition platform. **(3), (6)** Live view of camera nodes with overlayed targets currently tracking. **(4,5)** Rendering of resulting person of (3) in XRT visualization system from two viewpoints. **(9-11)** More live renderings in XRT.

for hours on the raw surveillance footage. Our system is capable of tracking multiple persons in real time. The inter-camera tracking results are embedded as live textures in a consistent and geo referenced 3D world model acquired by a mobile platform. This enables the intuitive 3D visualization of tracking results decoupled from sensor views. Future research includes person identification using RFID tags, long term experiments and the acquisition of enhanced 3D models.

Acknowledgments

We would like to thank Matrix Vision for their generous support and successful cooperation and Michael Wand for providing the XRT rendering framework.

References

[1] "What happens in vegas stays on tape," http://www.csoonline.com/read/090105/hiddencamera_vegas_3834.html. 1

[2] "Surveillance works: Look who's watching," *IEEE Signal Processing Magazine*, vol. 22, 3 2005. 1

[3] T. Boult, A. Lakshmikumar, and X. Gao, "Surveillance methods," in *Tutorial on IEEE Computer Vision and Pattern Recognition (CVPR)*, 2005. 1

[4] N. Siebel and S. Maybank, "The advisor visual surveillance system," in *ECCV 2004 workshop Applications of Computer Vision (ACV)*, 2004. 1

[5] M. M. Trivedi, T. L. Gandhi, and K. S. Huang, “Distributed interactive video arrays for event capture and enhanced situational awareness,” *IEEE Intelligent Systems, Special Issue on Homeland Security*, October/November 2005. 1

[6] D. B. Yang, H. H. González-Baños, and L. J. Guibas, “Counting people in crowds with a real-time network of simple image sensors.” in *ICCV*, 2003, pp. 122–129. 1

[7] 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. 2

[8] M. Bramberger, A. Doblander, A. Maier, B. Rinner, and H. Schwabach, “Distributed embedded smart cameras for surveillance applications,” vol. 39, 2006. 2

[9] W. Wolf, B. Ozer, and T. Lv, “Smart cameras as embedded systems,” *Computer*, vol. 35, no. 9, pp. 48–53, 2002. 2

[10] A. Doucet, N. D. Freitas, and N. Gordon, *Sequential Monte Carlo Methods in Practice*. Springer Verlag, 2001. 2

[11] M. Isard and A. Blake, “Condensation – conditional density propagation for visual tracking,” *International Journal of Computer Vision*, vol. 29, no. 1, 1998. 2

[12] “Special issue on: Sequential state estimation: From kalman filters to particle filters,” *Proceedings of the IEEE*, vol. 92, no. 3, 2004. 2

[13] D. Comaniciu, V. Ramesh, and P. Meer, “Kernel-based object tracking,” *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 25, no. 05, pp. 564–575, 2003. 2

[14] K. Okuma, A. Taleghani, N. de Freitas, J. J. Little, and D. G. Lowe, “A boosted particle filter: Multitarget detection and tracking,” in *ECCV 2004: 8th European Conference on Computer Vision*, 2004. 2

[15] P. Prez, C. Hue, J. Vermaak, and M. Gangnet, “Color-based probabilistic tracking,” in *European Conference on Computer Vision, ECCV’2002*, Copenhaguen, Denmark, June 2002, pp. 661–675. 2

[16] “Matrix vision,” http://www.matrix-vision.com/. 2

[17] S. Fleck and W. Straßer, “Adaptive probabilistic tracking embedded in a smart camera,” in *IEEE CVPR Embedded Computer Vision Workshop*, 2005. 3, 4

[18] S. Fleck, S. Lanwer, and W. Straßer, “A smart camera approach to real-time tracking,” in *13th European Signal Processing Conference (EUSIPCO)*, September 4-8 2005. 3, 4

[19] P. Biber, S. Fleck, M. Wand, D. Staneker, and W. Straßer, “First experiences with a mobile platform for flexible 3d model acquisition in indoor and outdoor environments – the wägele,” in *3D-ARCH*, 2005. 5, 6

[20] S. Fleck, F. Busch, P. Biber, H. Andreasson, and W. Strasser, “Omnidirectional 3d modeling on a mobile robot using graph cuts,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2005. 6

[21] “Project’s website,” http://www.gris.uni-tuebingen.de/~sfleck/smartsurv3d/. 6