ViRoom — Low Cost Synchronized Multicamera System and its Self-Calibration

Tomáš Svoboda, Hanspeter Hug, and Luc Van Gool

Computer Vision Laboratory
Department of Information Technology and Electrical Engineering
Swiss Federal Institute of Technology
Gloriastrasse 35, 8092 Zürich, Switzerland
svoboda@vision.ee.ethz.ch, http://www.vision.ee.ethz.ch/

Abstract. This paper presents a multicamera Visual Room (ViRoom). It is constructed from low-cost digital cameras and standard computers running on Linux. Software based synchronized image capture is introduced. A fully automatic self-calibration method for multiple cameras and without any known calibration object is proposed and verified by 3D reconstruction experiments. This handy calibration allows an easy reconfiguration of the setup. Aside from the computers which are usually already available, such a synchronized multicamera setup with five or six cameras costs less than 1000 USD.

1 Introduction

With decreasing prices of powerful computers and cameras smart multicamera systems start to emerge. Some of them were developed for creating realistic 3D models or mixing real humans with artificial data, (virtualized reality [5]). In late 1990's, new kinds of multicamera setups appeared. The main goal was not so much reconstruction but action recognition and interpretation. Cameras were often combined with microphone arrays for voice recognition and/or navigation. The EasyLiving project from Microsoft Research is developing an intelligent room that will be able to unobtrusively interact with a user [1]. The AVIARY project uses cameras and a microphone array for finding moving and/or speaking person. The detected speaker is then tracked and the best camera is chosen. A limited number of basic events are recognized [12]. The monitoring of a person's activities in the Intelligent meeting room, is the second part of that project. Detected activities are annotated and stored in a database for future browsing [8]. However, as stated in the overview [9], the interpretation of the activities is only at an early stage of development. Only a small subset of the rich variety of human gestures can be recognized.

Our work is similar to the AVIARY project. However, the cameras used in most of the previous projects are expensive, with external synchronization. Such a multicamera setup requires additional hardware, is restricted to one room and is hard to reconfigure. We wanted to build an easily reconfigurable and scalable environment without unnecessary cabling. We use simple digital cameras with an

IEEE1394 (Firewire) interface. The current IEEE1394 standard has 400 Mbit/s bandwidth but a higher bandwidth is expected in the near future[1]. The simple cameras do not allow external synchronization. We propose a software based synchronized capture.

A multicamera system can be efficiently used even without calibration, as shown e.g. in recent multicamera tracking work [6]. Nevertheless, a calibrated system has many advantages, like for 3D human motion analysis, inserting virtual objects into video streams, or aligning of the camera system with a previously reconstructed scene.

The proposed self-calibration method utilizes state of the art methods. It is mainly based on the self-calibration scheme proposed by Pollefeys et al. [10]. It requires no calibration object. A person waves a standard laser pointer which is dimmed down by a piece of paper while captured by cameras. The positions of the laser are detected in each image with subpixel precision. As the laser is moved around and visits several points, these positions make up the projection of a virtual object (3D point cloud) with unknown 3D position. This merged image is the input of the self-calibration procedure.

2 Setup and synchronization of cameras

The main underlying ideas of the setup can be characterized as follows:

- Use of many simple and *cheap cameras* rather than a few complicated and expensive pan and tilt devices.
- The system should allow easy and frequent *reconfiguration*, like adding, removing or replacing cameras.
- It should *adapt* its functionality to the number of available cameras and their arrangement. A slight mobile version with few cameras is also foreseen.
- Synchronized, or at least almost synchronized, capture is essential for consistent multicamera analysis.

2.1 Camera setup

We decided to use digital cameras with IEEE1394 (Firewire) interface and PCs running on Linux. Up to three cameras can be connected with a Linux machine and processed simultaneously. The Linux programming environment offers standards for network computing.

One overhead camera with very wide field of view gives an overview of the scene. It helps to resolve ambiguities, especially in arrangements with few cameras and many occlusions. The purchased cameras can be seen in Fig 2.

[1] http://www.1394ta.org/Technology/About/1394b.htm

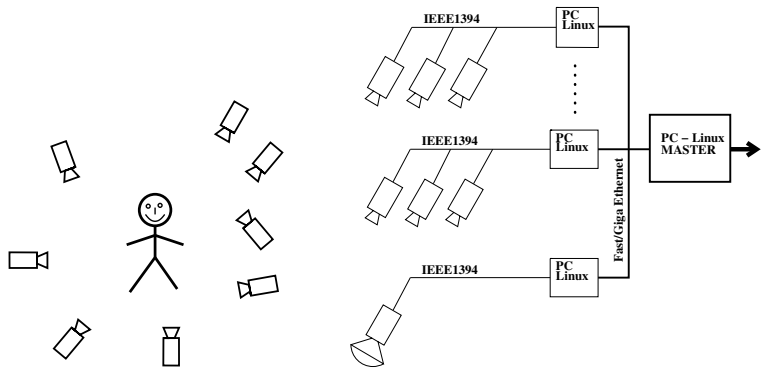

Fig. 1. Cameras may be placed relatively arbitrary, but a significant overlap of their fields of view is necessary. Cameras are connected to PCs by Firewire cables.

Fig. 2. Left: PYRO 1394 WebCam from ADS Technologies is a simple, cheap (around 100 USD) color digital camera. Right: SONY DFW-V500 is a more advanced digital camera with C-mount. The photo shows it with a fish-eye lens. Both cameras have a resolution of 640×480 pixels and allow progressive scan (non-interlaced) at 30 fps.

2.2 Software based synchronization

The simple cheap cameras do not allow synchronization through external triggering. More advanced digital cameras, like the one we use with the fish-eye lens, allow this however, cost approximately 15 or 20 times more. Moreover, external triggering would require additional hardware and cabling which might be acceptable for a fixed environment but would be obtrusive for a mobile version. Therefore, we propose a software based synchronization.

The *CameraD*[2] system consists of two parts. The camera server and various clients. The camera server runs on each computer that has one or more cameras attached to it. The clients can run on any machine with a network connection.

The *Camerad* server is a rather simple internet server. After startup it waits for client connections. When a connection arrives, *Camerad* stops listening for further connections in favor of servicing the one already established. As a con-

[2] From CAMERA Daemon.

sequence only one client can connect to any given camera server at any point in time. This "one client only" rule is rather unusual for internet servers but it is no particular disadvantage in our application. It simplifies the camera server in two ways. First, there is no need for the camera server to sequentialize simultaneous access to the hardware (the cameras) by multiple clients. Second, the camera server does not need to service a network connection while it accesses the hardware. Capturing is controlled by a single client. The client opens a connection to each camera server and then sends a "trigger" signal simultaneously to all servers. After receiving this signal, the servers will immediately start capturing an image. The timing accuracy depends on:

- How fast the client can send signals to multiple camera servers. With an unloaded ethernet connection the time between the signal to the first server and the signal to the last server is reasonably short ($\leq$ 1ms). However, if the network is loaded or if a different type of network is used (e.g. a dialup modem connection) this delay might be unpredictably long.
- How fast the servers can react to a received signal. This response time is dictated by the Linux scheduler. On a normal Linux system time slices are 10 ms long which means that the response time will be $(10n+m)$ [ms] where n is a non-negative integer and $0.0 \leq m \leq 10.0$. As system load goes up n will increase and the expected value for m will approach 5.0. Running the camera server with realtime priority should reduces n to zero and m close to it. (e.g. through the use of the low latency patch which is expected to become part of standard Linux). This version has not been tested yet.
- The camera drivers and hardware, these aspects are not controlled by the camera server. Currently it is assumed that the delays introduced by the camera drivers and hardware are the same on all systems involved.

Our current implementation of the *CameraD* package contains a client called "cdcap"[3] which captures synchronously images from different cameras on different computers and stores them on local disks. The implementation and the firewire library are not mature yet, hence, only up to 10 n-tuple synchronized images can be captured per second. Still, it is enough for first self-calibration, motion segmentation and tracking experiments.

3 Self-calibration

3.1 Theory

Let us consider n cameras and m object points $\mathbf{X}_j = [X_j, Y_j, Z_j, 1]^T, j = 1, \ldots, m$. We assume the pinhole camera model is valid, see [4] for details. The points $\mathbf{X}_j$ are projected to image points $\mathbf{u}_{ij}$ as

$$\lambda_{ij} \begin{bmatrix} u_{ij} \\ v_{ij} \\ 1 \end{bmatrix} = \lambda_{ij} \mathbf{u}_{ij} = P_i \mathbf{X}_j \,, \qquad \lambda_{ij} \in \mathbb{R}^+ \tag{1}$$

[3] CameraD CAPture

where each P_i is a 3×4 matrix that contains 11 camera parameters. They depend on the six DOF that describe camera position and orientation and on the five internal parameters. The $\mathbf{u}_{ij}$ are observed pixel coordinates. The goal of the calibration is to estimate scales λ_{ij} and the camera projection matrices P_i. We

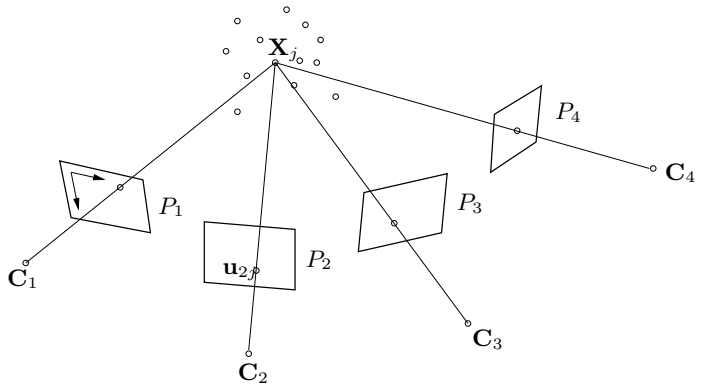

Fig. 3. Multicamera setup with 4 cameras.

can put all (1) into one matrix W_s:

$$W_s = \hat{P}\hat{X}\,, \tag{2}$$

where W_s is called the *scaled measurement matrix*, $\hat{P} = [P_1 \cdots P_n]^\top$ and $\hat{X} = [\mathbf{X}_1 \cdots \mathbf{X}_m]$. $\hat{P}$ and $\hat{X}$ are referred to as the *projective motion* and the *projective shape*. If we collect enough noiseless points (u_{ij}, v_{ij}) and the λ_{ij} are known, then W_s has rank 4 and can be factorized into $\hat{P}$ and $\hat{X}$. The factorization (2) recovers the motion and shape up to a 4×4 linear projective transformation H:

$$W_s = \hat{P}\hat{X} = \hat{P}HH^{-1}\hat{X} = PX, \tag{3}$$

where $P = \hat{P}H$ and $X = H^{-1}\hat{X}$. Any non-singular 4×4 matrix may be inserted between $\hat{P}$ and $\hat{X}$ to get another compatible motion and shape pair P, X. The self-calibration process computes such a matrix H that P and X become Euclidean. The task of finding the appropriate H can only be achieved by imposing certain geometrical constraints. The most general constraint is the assumption of orthogonal rows and columns of the CCD chips in cameras. Alternatively, we can assume that some internal parameters of the cameras are the same, which would be more useful for a monocular camera sequence. The minimal number of cameras for successful self-calibration depends on the number of known camera parameters or the number of fixed parameters. For instance, 8 cameras are needed when the orthogonality of rows and columns is the only constraint and three cameras are sufficient if all principal points are known or if the internal camera parameters are completely unknown but the same for all cameras. See [4] for a more detailed treatment of self-calibration theory.

3.2 Practical implementation

The main problem in the self-calibration is usually establishing a set of correspondences $\mathbf{u}_{ij}$. We exploit the synchronized capture. A person moves a standard laser pointer around, which is dimmed down by a piece of paper and is seen by the cameras. The very bright projections of the laser can be detected in each image with subpixel precision by fitting an appropriate point spread function. These particular positions are then merged together over time, creating thus projections of a virtual 3D object. Our proposed self-calibration scheme can be outline as follows:

1. Find the projections of the laser pointer in the images.
2. Discard wrongly detected points or incorrect matches made due to imperfect synchronization by pairwise RANSAC analysis [2].
3. Estimate projective depths λ_{ij} from (2) by using the approach proposed in [11].
4. Perform the rank 4 factorization of the matrix W_s to get projective shape and motion [4].
5. Upgrade the projective structures to Euclidean [10, 3].

Our current implementation of the factorization method requires to have correspondences across all images. This would be a significant constraint for future installations of the Visual Room. However, we plan to use a new method proposed in [7] which was successfully tested on image sets where a significant portion of correspondences was missing.

4 Experiments

Our first Visual Room consists of four computers and four cameras, one camera attached to one computer. Its arrangement and an example of the 4 camera views is shown in Fig 4.

We performed several self-calibration experiments. We put the principal points of the cameras in the center of the images and assume orthogonality of CCD rows and columns. No other information about the cameras is used. We evaluate the reliability and precision of the self-calibration by measuring 2D reprojection errors and by performing 3D reconstruction of points on the two visible walls. A person waved a laser pointer and about 150 image 4-tuples were captured. From these data, 97 4-correspondences survived the RANSAC validation step. Calibration results can be seen in Fig 5. The average reprojection error is 0.69 pixels with a standard deviation of 0.41 pixels. It is difficult to directly evaluate the self-calibration results – the camera matrices P_i. We can do it by reconstructing a known 3D object. We pointed the laser pointer to two visible walls to generate virtual points. Correspondences were established by the same method as for the self-calibration. The camera matrices P_i were inserted into a linear triangulation method which minimizes the algebraic error [4] of 3D reconstruction. The reconstructed points are shown in 5. A linear method was used

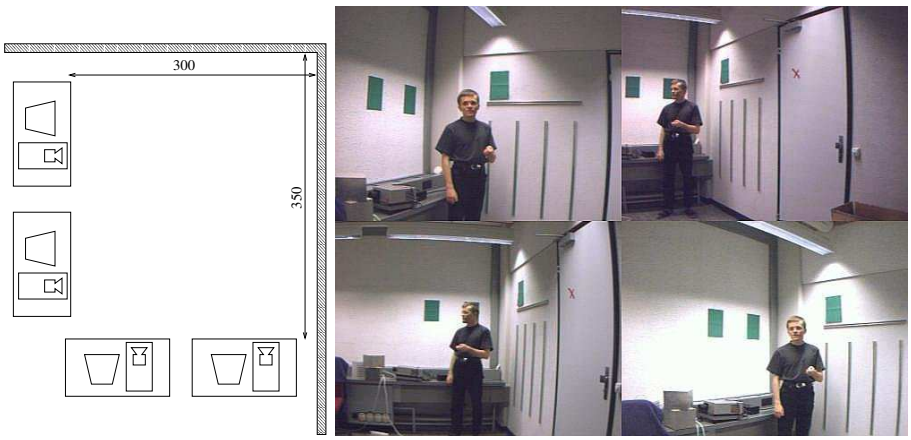

Fig. 4. Left: A sketch of our first ViRoom. Right: Images captured by our first prototype of Visual Room.

to find the parameters of the two planes. The angle between the planes is 92°. We did several calibration experiments and the angle between the reconstructed walls varied in the range $90 \pm 5°$.

5 Conclusion

The ViRoom, a low cost synchronized multicamera system, was introduced. It consists of standard PCs running on Linux and simple digital cameras with firewire interfaces. No other special hardware is required. A software package for synchronized capturing based on the TCP/IP communication protocol was developed. The proposed self-calibration method does not require a special calibration object. The only input in this calibration is a sequence of images with easily detectable bright spots.

We are currently working on the synchronized image capture, with a larger set of cameras, on self-calibration with occlusions to some of the cameras and on the integration of this multicamera system with our body modeling and gesture analysis software.

Acknowledgment

Support from the ETH project "Blue-C" is gratefully acknowledged. We thank Ondřej Chum from CMP at CTU Prague for his implementation of the RANSAC.

References

1. Barry Brumitt, Brian Meyers, John Krumm, Amanda Kern, and Steven Shafer. Easyliving: Technologies for intelligent environments. In *Proceedings of the 2nd International Symposium on Hanheld and Ubiquitos Computing*, pages 12–29, September 2000.

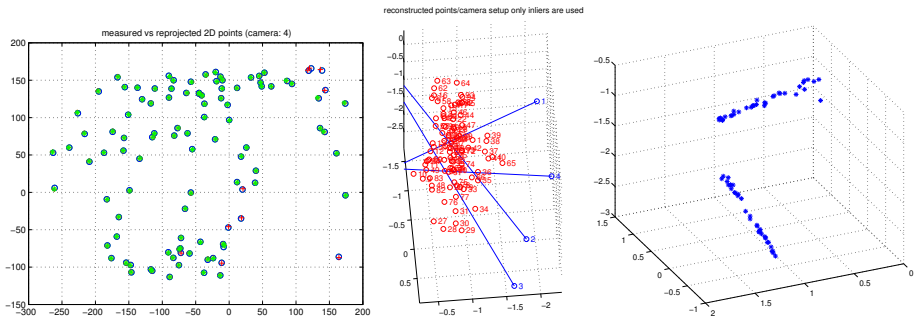

Fig. 5. Calibration results. Left: Positions of the laser pointer are denoted by small circles. Green asterisks denote reprojected good points and red crosses reprojection of points which did not come through the RANSAC validation. Middle: Reconstructed points, positions of cameras and orientations of optical axes. Right: Reconstructed virtual wall points.

2. M.A. Fischler and R.C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, June 1981.
3. Mei Han and Takeo Kanade. Creating 3D models with uncalibrated cameras. In *Proceeding of IEEE Computer Society Workshop on the Application of Computer Vision (WACV2000)*, December 2000.
4. R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, Cambridge, UK, 2000.
5. Takeo Kanade, P.J. Narayanan, and Peter W. Rander. Virtualized reality: Concepts and early results. In *IEEE Workshop on the Representation of Visual Scenes*, pages 69–76, June 1995.
6. Sohaib Khan, Omar Javed, Zeeshan Rasheed, and mubarak Shah. Human tracking in multiple cameras. In *International Conference on Computer Vision*, July 2001.
7. Daniel Martinec and Tomáš Pajdla. Structure from many perspective images with occlusions. In *European Conference on Computer Vision*. Springer-Verlag, May 2002. To appear.
8. Ivana Mikic, Koshia Huang, and Mohan Trivedi. Activity monitoring and summarization for an intelligent room. In *IEEE Workshop on Human Motion*, pages 107–112, December 2000.
9. Alex Pentland. Looking at people: Sensing for ubiquitous and wearable computing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1):107–119, January 2000.
10. Mark Pollefeys, Reinhard Koch, and Luc Van Gool. Self-calibration and metric reconstruction inspite of varying and unknown intrinsic camera parameters. *International Journal of Computer Vision*, 32(1):7–25, August 1999.
11. Peter Sturm and Bill Triggs. A factorization based algorithm for multi-image projective structure and motion. In *European Conference on Computer Vision*, pages 709–720. Springer - Verlag, 1996.
12. Mohan M. Trivedi, Ivana Mikic, and Sailendra K. Bhonsle. Active camera networks and semantic event databases for intelligent environments. In *IEEE Workshop on Human Modeling, Analysis and Synthesis (in conjunction with CVPR)*, June 2000.