Multimedia Systems (2006) 12:195–210
DOI 10.1007/s00530-006-0057-6

REGULAR PAPER

Calibrating and optimizing poses of visual sensors in distributed platforms

Eva Hörster · Rainer Lienhart

Published online: 10 October 2006

Abstract Many novel multimedia, home entertainment, visual surveillance and health applications use multiple audio-visual sensors. We present a novel approach for position and pose calibration of visual sensors, i.e., cameras, in a distributed network of general purpose computing devices (GPCs). It complements our work on position calibration of audio sensors and actuators in a distributed computing platform (Raykar et al. in proceedings of ACM Multimedia '03, pp. 572–581, 2003). The approach is suitable for a wide range of possible – even mobile – setups since (a) synchronization is not required, (b) it works automatically, (c) only weak restrictions are imposed on the positions of the cameras, and (d) no upper limit on the number of cameras under calibration is imposed. Corresponding points across different camera images are established automatically. Cameras do not have to share one common view. Only a reasonable overlap between camera subgroups is necessary. The method has been sucessfully tested in numerous multi-camera environments with a varying number of cameras and has proven itself to work extremely accurate. Once all distributed visual sensors are calibrated, we focus on post-optimizing their poses to increase coverage of the space observed. A linear programming approach is derived that determines jointly for each camera the pan and tilt angle that maximizes the coverage of the space at a given sampling frequency. Experimental results clearly demonstrate the gain in visual coverage.

Categories and Subject Descriptors

I.4.10 [Image Processing and Computer Vision]: General; I.5.4 [Pattern Recognition]: Applications

General Terms

Algorithms

Keywords Multi-camera calibration · Sensor networks · Position calibration · Pose optimization

E. Hörster (✉) · R. Lienhart
Multimedia Computing Lab, University of Augsburg, Augsburg, Germany
e-mail: hoerster@informatik.uni-augsburg.de

R. Lienhart
e-mail: lienhart@informatik.uni-augsburg.de

1 Introduction

Today we can find microphones, cameras, loudspeakers and displays nearly everywhere - in public, at home and at work. These audio/video sensors and actuators are often a component of computing and communication devices such as laptops, PDAs and tablets, which we refer to as General Purpose Computers (GPCs). Often GPCs are networked using high-speed wired or wireless connections. The resulting array of audio/video sensors and actuators along with array processing algorithms offers a set of new features for multimedia applications such as video conferencing, smart conference rooms, video surveillance, games, e-learning, home entertainment and image based rendering.

Many of the above mentioned audio-visual array processing algorithms require precise knowledge about the positions and poses of the sensors and actuators as well as the coverage that is achieved by those sensors. This demands a simple and convenient calibration approach to put all sensors and actuators into a common time and space. Lienhart et al. [14] propose a means to provide a common time reference to multiple distributed GPCs. In [21] a method for automatically calibrating audio sensors and actuators is presented. In this paper we focus on visual sensors where a room or area is instrumented with $N \geq 3$ static cameras connected to networked GPCs. No precise synchronization of the different devices is required.

In the first part of this paper we focus on providing a common space for multiple cameras by actively estimating their 3D positions and poses. We also address the problem of effortlessly calibrating the intrinsic parameters of multiple cameras.

In the second part of the paper another important issue in designing visual sensor arrays is considered: orienting the visual sensors such that they achieve optimal coverage of a given space at a predefined 'sampling rate' (see Sect. 3 for a precise definition). We assume that the positions and inital poses are given. This is reasonable because either cameras have been already installed (e.g., at an airport), or they are put up arbitrarily. Currently there exists only few theoretical research on planning visual sensor positions and poses. Positions and inital poses of the multiple cameras can be determined automatically by our calibration approach (see Sect. 2). Given the fixed positions, we develop a linear programming model that determines the optimal poses (pan and tilt angles) with respect to coverage while maintaining the required resolution (i.e., minimal 'sampling frequency'). Figure 1 shows one ineffective setup that we desire to optimize.

Related work: Camera calibration is a well researched topic in computer vision. Fundamentally there are two different methods of camera calibration: photogrammetric calibration and self-calibration [30]. The first method uses a 3D, a 2D (planar), or a virtual calibration object of precisely known geometry. Important approaches are described in [4,11,26,27,30]. Planar methods are very popular because it is easy to obtain a calibration target by just printing the pattern and fixing the paper on a flat surface. Although providing good results, the major drawback of these calibration methods is that they require special equipment or precise manual measurements. Virtual calibration objects are constructed over time by tracking an easily identifiable

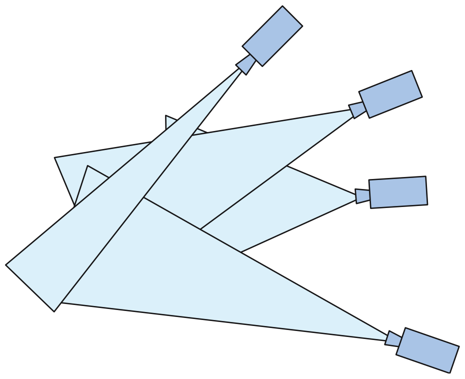


Fig. 1 Example of an inefficient setup we desire to optimize

object through a 3D scene. The cameras usually have to be synchronized and thus the setup requires special equipment. Self calibration techniques [9,19,25] do not require any special calibration target. They simultaneously process several images from different perspectives of a scene and are based on point correspondences across the images. The accuracy of these methods depends on how accurately those point correspondences can be extracted between images. Point correspondences are extracted automatically from the images by identifying 2D features and tracking those between the different perspective views. Different feature extraction algorithms exist (see [8,15,23]). There exist also self-calibration approaches using silouettes or trajectories of moving objects [20,24]. Multiple camera calibration can be solved globally in one step, or multiple subsets of cameras and displays are calibrated first and then merged into a global coordinate system. Since the first method is only suitable if all cameras share a common view, we follow the second more general approach.

Although a significant amount of research exists in designing and calibrating video sensor arrays, automated visual sensor placement and alignment in general has not been addressed often. There is some work in the area of grid coverage problems with sensors sensing events that occur within a distance *r* (the sensing range of the sensor) [3,22,28,31]. Our work is based on those approaches, but differs in the sensor model (since cameras do not posses circular sensing ranges) as well as the cost function and some constraints. In [5] a camera placement algorithm based on a binary optimization technique is proposed. The algorithm aims to find the placement with minimum cost of a camera set such that a given space is viewed with some minimal spatial resolution. Space is represented as a occupancy grid and the authors focused on planar regions. A similar task is considered in [12] and also solved by linear programming techniques. In [18] the authors analyze the visibility from static sensors probabilistically and present a solution for maximizing visibility in a given region of interest. They solve the problem by simulated annealing.

Contributions: The main contributions of the paper are:

- A procedure to automatically calibrate the positions and poses of sensors without using calibration objects. Thus no special equipment is required. In addition the setup does not have to be synchronized. It only requires to filter out temporally unstable salient points and keep only stationary features. Our method is simple and convenient to use and offers mobility of the entire setup. The camera views are assumed to overlap only partly, i.e., only some cameras share a common view.
- The usage of an active display as our calibration target for intrinsic calibration giving us control over the calibration pattern to be displayed. As a result the extraction of feature points is easier and more reliable. The calibration pattern can be made adaptive to the distance between the camera and the pattern's image on the LCD screen.
- The automatic extraction of control points and point correspondences across images.
- A procedure to determine the optimal poses of the cameras such that coverage is maximized while maintaining a minimal resolution.

The rest of the paper is organized as follows. In Sect. 2 we formulate the calibration problem and present our solution. We describes how point features are extracted and tracked between images and outline the calibration of the intrinsic parameters of each camera. The algorithm used to determine the extrinsic parameters, i.e., the positions and poses of all cameras in a common coordinate system is presented. In Sect. 3 we formulate the optimization problem of maximizing coverage with multiple cameras by pose variation. Our solution is presented and results are reported. The paper concludes with a summary and an outlook in Sect. 4.

2 Mulitple camera calibration

2.1 Problem formulation

Given M cameras, the goal is to determine the cameras' internal parameters and the 3D positions and poses of the cameras automatically. Therefore we only make the assumption, that we know the number of visual sensors in the network.

In this work we use an enhanced perspective model to describe our cameras. The mapping performed by a perspective camera between a 3D point $\mathbf{X}$ and its 2D image point $\mathbf{x}$, both represented by their homogeneous coordinates, is usually represented by a 3×4 matrix, the camera projective matrix $\mathbf{P}$: $\mathbf{x} \simeq \mathbf{PX}$. The matrix $\mathbf{P}$ can be written as $\mathbf{P} = \mathbf{K}[\mathbf{R}|\mathbf{t}]$ where $\mathbf{K}$ is a 3×3 upper triangular matrix containing the camera intrinsic parameters:

$$\mathbf{K} = \begin{pmatrix} f_x & s & p_x \\ 0 & f_y & p_y \\ 0 & 0 & 1 \end{pmatrix} \tag{1}$$

The parameters f_x and f_y denote the focal length, p_x and p_y denote the coordinates of the principal point, each in terms of pixel dimensions. s denotes the skew. For most commercial cameras, and hence below, the skew is considered to be zero. The 3×3 rotation matrix $\mathbf{R}$ and the 3×1 translation vector $\mathbf{t}$ describe the 3D position and pose of the camera. As some desktop cameras exhibit significant distortions, this model has to be enriched by some distortion components. The distortion model introduced in [11] accounts for tangential and radial distortions using two coefficients. It describes distortions occuring in practice sufficiently precise. In the following discussion we assume that the distortion parameters of each camera are known and the effects of those have been removed from all images.

Different views of the same scene are related to each other. These relations can be used for our multiple camera calibration task. Therefore we need to determine a set of corresponding points across the different images. Points are said to correspond if they represent the same scene point in different views. This general calibration problem is illustrated in Fig. 2.

A set of 3D points $\mathbf{X}_i$ is viewed by a set of cameras with matrices $\mathbf{P}^j$. Let $\mathbf{x}_i^j$ denote the coordinates of the i-th point as detected in the j-th camera image.

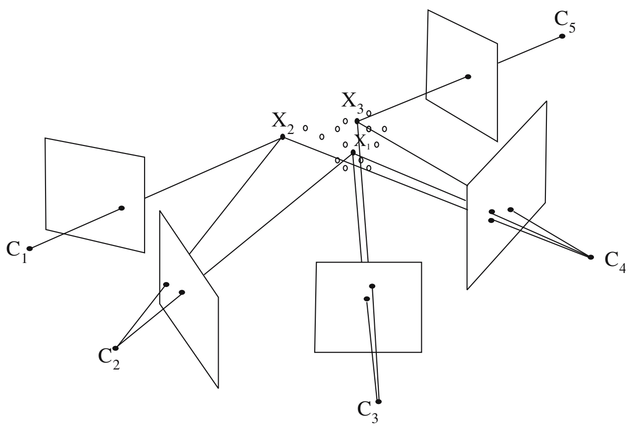


Fig. 2 General calibration problem

A 3D point may not be visible in all cameras, thus its corresponding projected point will not be available in all images. The calibration problem is then to find the set of camera matrices $\mathbf{P}^j$ and points $\mathbf{X}_i$ such that for all image points $\mathbf{x}_i^j \simeq \mathbf{P}^j\mathbf{X}_i$ holds. However, unless additional constraints are given, it is in principle only possible to determine the camera matrices up to a projective ambiguity. Additional constraints arising from knowledge about the cameras' parameters and/or the scene can be used to restrict this ambiguity up to an affine, metric or Euclidean transformation.

Solution: We solve the camera calibration problem in two stages. In a first stage we determine the cameras' intrinsic parameters. Intrinsic calibration is done independently for each camera by using a flat-panel display as the planar calibration object. In a second stage camera positions and poses are computed in a common coordinate system (extrinsic calibration). Their positions and poses can be determined relative to each other up to a global scale factor. In a typical distributed camera environment each camera can only see a small volume of the total viewing space and different intersecting subsets of cameras share different intersecting views. Hence multiple camera calibrations are performed by calibrating subsets of cameras and then building a global coordinate system from individual overlapping views.

2.2 Point correspondences

2D point correspondences between projections of the same 3D point onto different camera planes can be generally used to recover the calibration matrices of the cameras. Therefore establishing such correspondences is the first step in determining the cameras parameters. To establish point correspondences each image is at first represented by a set of features. Each feature describes a specific image point, and its neighborhood. Subsequently these features are input to a matching procedure, which identifies features in different images that correspond to the same point in the observed scene. There are various approaches for extracting a set of interest points and features from an image. Our approach uses the so called SIFT-features proposed in [15]. SIFT-based feature descriptors were identified in [17] to deliver the most suitable features in the context of matching points of a scene under different viewing conditions such as different lighting and changes in 3D viewpoint.

SIFT-features extraction: The SIFT-feature extraction method combines a scale invariant region detector and a descriptor based on the gradient distribution in the detected regions. In order to compute a set of caracteristic image features, first a set of interest points – also called keypoints – is found by detecting scale-space extremas. Only keypoints that are stable under a certain amount of additive noise are preserved. An image location, scale and orientation is assigned to each keypoint. This enables the construction of a repeatable local 2D coordinate system, in which the local image (pixel and its surrounding region) is described invariantly from these parameters. Finally a descriptor for each keypoint is calculated based upon image gradients in the local image. However this approach has its limitations. To ensure a sufficient number of reliable matching points, the displacement between the cameras should not exceed 15°. The resulting correspondences are within pixel accuracy.

SIFT-feature matching: The matching technique used for the SIFT-features has been proposed in [15]. Point correspondences between two images are established by comparing their respective keypoint descriptors. Matching is performed by first individually measuring the Euclidean distance of each feature vector (representing a certain keypoint) of one image to each feature vector of the other image. The best matching candidate for a specific keypoint is identified by the keypoint belonging to the feature vector with the minimum distance. A match is found in the second image, if the distance ratio between the nearest and the second nearest neighbor (closest/second closest) is below a threshold. An example of matched points between two images is shown in Fig. 3.

Subpixel accuracy: The result of SIFT-feature matching is only at pixel accuracy. For position estimation of multiple cameras experiments have shown that it is essential to keep all computations at a subpixel accuracy level. So far the approximate positions of the corresponding points are known. To achieve subpixel precision we use the Affine Lucas Kanade feature tracker [23]. This tracker assumes an affine transformation between the viewpoint of both images. This approximation is valid, if the viewpoints of the different cameras are sufficiently close.

The basic optimization problem solved by the feature tracker is:

$$\min_{\mathbf{d},\mathbf{D}} \sum_{x=-\omega_x}^{\omega_x} \sum_{y=-\omega_y}^{\omega_y} (I(\mathbf{x}+\mathbf{u}) - J((\mathbf{D}+\mathbf{I}_{2\times 2})\mathbf{x}+\mathbf{d}+\mathbf{u}))^2 \tag{2}$$

where $I(\mathbf{u})$, $J(\mathbf{u})$ represent the grey-scale values of the two images at location $\mathbf{u}$, the vector $\mathbf{d} = [d_x\ d_y]^{\mathrm{T}}$ is the optical flow at location $\mathbf{u}$, and the matrix $\mathbf{D}$ denotes an affine deformation matrix characterized by the four coefficients $d_{xx}, d_{xy}, d_{yx}, d_{yy}$:

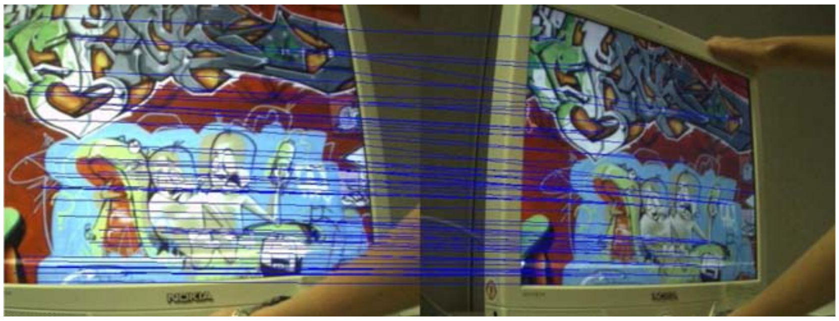


Fig. 3 Matched points are visualized by a connecting line between images

$$\mathbf{D} = \begin{pmatrix} d_{xx} & d_{xy} \\ d_{yx} & d_{yy} \end{pmatrix} \quad (3)$$

The objective of affine tracking is then to choose $\mathbf{d}$ and $\mathbf{D}$ in a way that minimizes the dissimilarity between feature windows of size $2\omega_x + 1$ in x and size $2\omega_y + 1$ in y direction around the point $\mathbf{u}$ and $\mathbf{v}$ in I and J respectively. $\mathbf{v}$ denotes the corresponding point to $\mathbf{u}$ and can be expressed in terms of $\mathbf{u}$ according to $\mathbf{v} = \mathbf{u} + \mathbf{D}\mathbf{u} + \mathbf{d}$. To handle changes in brightness and contrast a normalization is applied to the image patches in the iteration process.

An example result obtained by the subpixel feature tracking algorithm is shown in Fig. 4. The corresponding points obtained by SIFT-feature matching were used to initialize the algorithm. They are shifted to a slightly different position by the tracker. The improvement in accuracy is especially obvious in the region marked with a circle in both images. SIFT-feature matching in the case of two very close points in the first image resulted in the same feature in the second image. With this initial guess the feature tracker regains the two different corresponding points and hence significantly improves the accuracy of the image matching process.

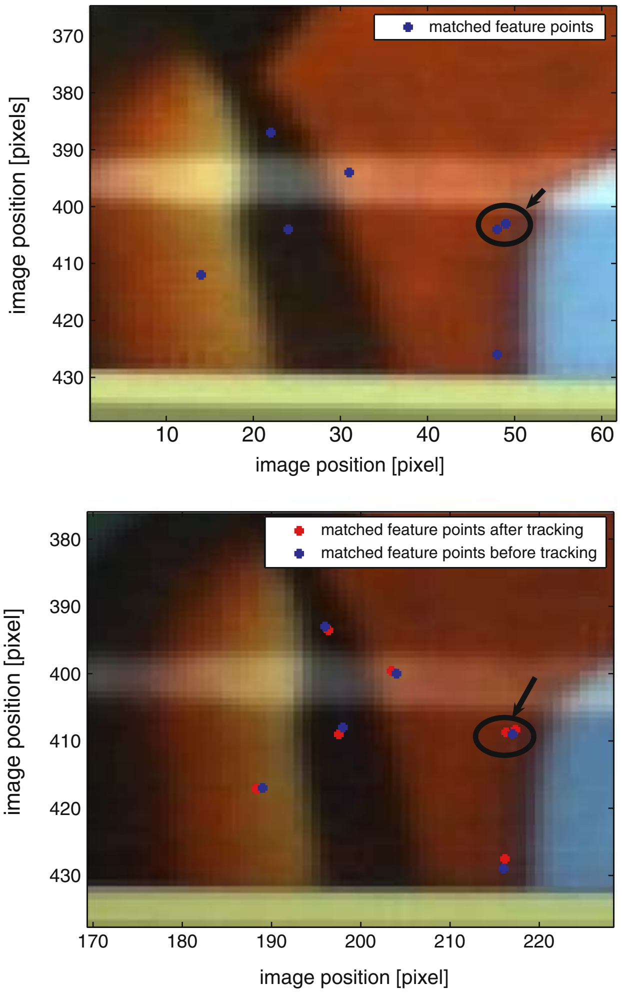


Fig. 4 Matched feature points before and after the tracking algorithm (points in the top image were taken as reference and tracked in the bottom image)

2.3 Intrinsic calibration

Intrinsic calibration is done independently for each camera using J.-Y. Bouguets Camera Calibration Toolbox [2] in OpenCV [14]. The calibration algorithm requires to record images of a known planar calibration target at a few (at least two) different orientations for each camera, where the motion of the different poses needs not to be known. Therefore the target can be freely moved in front of each camera separately. As the 2D geometry of the calibration plane is known with high precision, the camera observes in each image the projection of a set of control points with known position in some fixed world coordinate system. The Maximum Likelihood estimate of the camera parameters is obtained by minimizing the mean-squared distance between measured feature points in the image and their theoretical position with respect to the camera's intrinsic and extrinsic parameters, i.e., by minimizing the following error:

$$\varepsilon = \sum_{j=1}^{n} \sum_{i=1}^{m} \| \mathbf{x}_i^j - \hat{\mathbf{x}}(\mathbf{K}, \kappa_1, \kappa_2, \rho_1, \rho_2, \mathbf{R}_j, \mathbf{t}_j, \mathbf{X}_i) \|^2 \quad (4)$$

where n denotes the number of images taken of the model plane and m denotes the number of corresponding points each images gives rise to. $\hat{\mathbf{x}}(\mathbf{K}, \kappa_1, \kappa_2, \rho_1, \rho_2, \mathbf{R}_j, \mathbf{t}_j, \mathbf{X}_i)$ is the theoretical position of the projection of point $\mathbf{X}_i$ in the image j including distortion effects described by the distortion coefficients $\kappa_1, \kappa_2, \rho_1, \rho_2$. This is a nonlinear optimization problem which requires a proper initialization. Thus the complete calibration procedure consists of an initialization stage, where a closed form solution for the calibration parameters is computed, followed by a nonlinear refinement based on the Maximum Likelihood criterion. For more details on both stages the reader is reffered to [2,30].

Control point extraction: In the calibration procedure several different perspectives of a planar model object of known geometry are fed into the calibration routine. We used the pattern shown in Fig. 6 (right; from http://www.lear.inrialpes.fr/people/mikolajczyk) as our planar model object, since it gives rise to a large number of SIFT-features. It is displayed on a laptop or any other screen of known size, whose surface is sufficiently flat. Different images of the model plane from different orientations are captured by waving the screen in front of the camera. Some example images of the plane under different orientations are shown in Fig. 5. Projected pattern points in the images are determined by matching extracted SIFT-features from each view with extracted features from the calibration pattern. Subpixel matching was not necessary to obtain sufficiently accurate results. A matching example between the calibration pattern and an image of the model plane is illustrated in Fig. 6.

Usually in other calibration procedures [2,27] a checkerboard pattern is used as the calibration target requiring some user interaction to obtain matching points (in this case corners) between the this object and its (different) image(s). The use of SIFT-feature matching in combination with a flat screen displaying a known pattern enables us to easily and automaticaly detect the subset of image points. Additionally feature matching can be performed with images containing only a partially visible pattern.

Experimental results: In order to evaluate the calibration routine, the algorithm was applied to a different number of images of the model plane. The results are shown in Table 1. As the number of pattern points extracted varies per view, only the total number of points used in the calibration procedure is given for each experiment.

The influence of the number of images used for the calibration with respect to the performance of the optimization procedure was investigated in [30]. They found that the estimation error decreases with an increasing number of images of the model plane. This effect can also be observed in Table 1. For $n \geq 40$ images the estimated intrinsic parameters are consistent between the experiments, whereas in the case of only 20 views the calculated values show a relatively large deviation from these. Compared to algorithms using corner features, the number of views necessary for reliable calibration is larger, as corner correspondences can be extracted more accurately than SIFT feature correspondences. Once the intrinisc parameters are determined the distortion in the original images can be corrected as shown in Fig. 7.

2.4 Extrinsic calibration of multiple cameras

The main objective of the algorithm presented in this section is to recover the 3D positions and poses of multiple cameras in a common coordinate system in a fully automatic manner from the captured images of the different cameras. The considered situation is illustrated in Fig. 2. The mapping of a 3D point $\mathbf{X}_i$ to a 2D image point $\mathbf{x}_i^j$ can be described according to (assuming distortion effects have been removed):

$$\mathbf{x}_i^j \simeq \mathbf{P}^j \mathbf{X}_i \tag{5}$$

As a 3D point $\mathbf{X}_i$ might only be observed by a subset of cameras, the corresponding projected point will not appear in every view. Since we know the intrinsic parameters of all cameras in the scene, the locations of the projected points can be given in normalized image coordinates, denoted in the following with $\mathbf{x}_{\mathbf{n}_i}^j$. The normalized image coordinates $\mathbf{x_n}$ of a measured point $\mathbf{x}$ are derived by removing the effects of the internal parameters from the measured image point:

$$\mathbf{x_n} \simeq \mathbf{K}^{-1}\mathbf{x} \tag{6}$$

where $\mathbf{K}$ is the calibration matrix of the particular camera. The mapping between a point $\mathbf{X}_i$ to its projected and normalized point $\mathbf{x}_{\mathbf{n}_i}^j$ in the j-th image is then described by the normalized camera matrix $\mathbf{P}_{\mathbf{n}}^j$:

$$\mathbf{x}_{\mathbf{n}_i}^j \simeq \mathbf{P_n}^j \mathbf{X}_i \tag{7}$$

where

$$\mathbf{P}_{\mathbf{n}}^j = \mathbf{K}_j^{-1}\mathbf{P}^j = \mathbf{K}_j^{-1}\mathbf{K}_j[\mathbf{R}_j|\mathbf{t}_j] = [\mathbf{R}_j|\mathbf{t}_j] \tag{8}$$

The matrix $\mathbf{P_n}^j$ only consists of a rotation $\mathbf{R}_j$ and a translation $\mathbf{t}_j$, which define the position of the specific camera. Given a set of image correspondences, represented by their normalized coordinates $\mathbf{x}_{\mathbf{n}_i}^j$ our goal is now to find the appropriate set of normalized camera matrices $\mathbf{P}_{\mathbf{n}}^j$ and points $\mathbf{X}_i$ such that

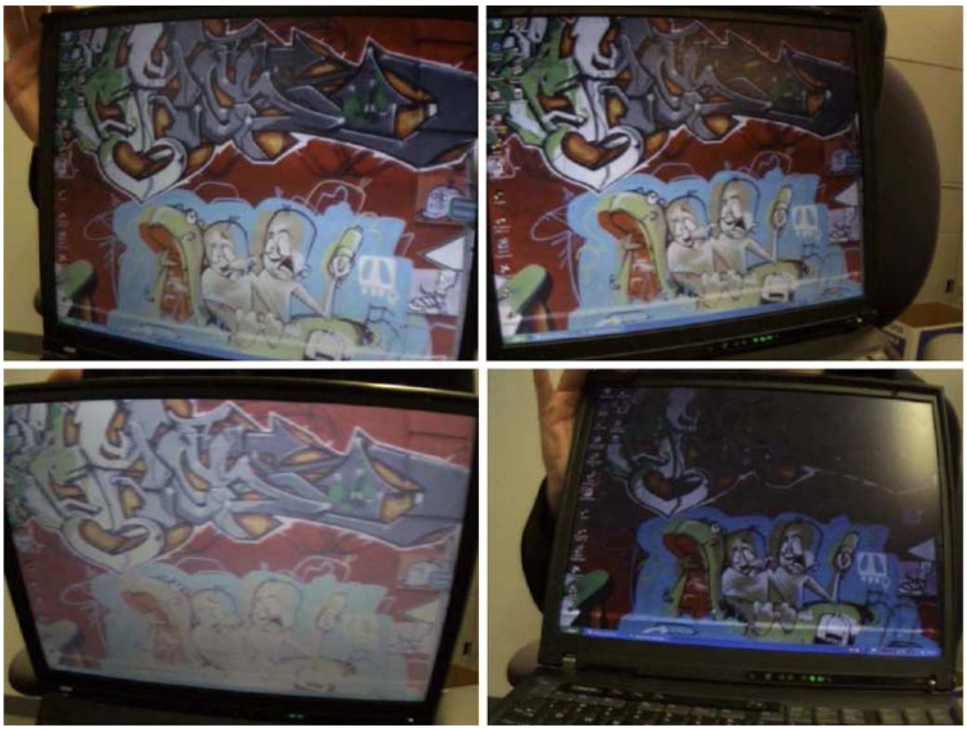


Fig. 5 Four sample images of the model plane used for calibration

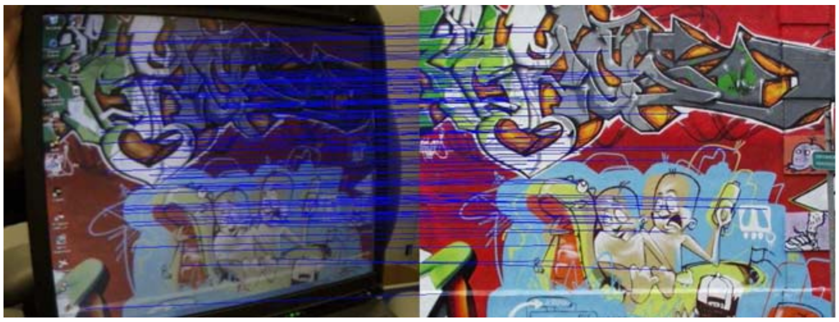


Fig. 6 Matches between calibration pattern *(right)* and its imaged version *(left)*

Table 1 Results obtained for the intrinsic parameters of a camera

# Images	20	30	40	50	60
# Points	2,218	3,735	5,171	6,735	8,039
$\mathbf{f}_x$	818.38	831.29	834.17	836.79	838.16
$\mathbf{f}_y$	818.16	830.87	833.64	836.38	837.98
$\mathbf{p}_x$	305.02	307.19	308.77	307.69	308.51
$\mathbf{p}_y$	263.87	257.35	255.35	255.32	254.48
κ_1	−0.421	−0.432	−0.437	−0.433	−0.436
κ_2	0.092	0.108	0.126	0.101	0.111
ρ_1	−0.005	−0.003	−0.002	−0.002	−0.002
ρ_2	0.004	0.003	0.002	0.003	0.003

$$\mathbf{x}_{\mathbf{n}_i}^j \simeq \mathbf{P}_\mathbf{n}^j \mathbf{X}_i \tag{9}$$

As the intrinsic parameters of each camera are known, the relative position of the cameras is computed uniquely up to an overall scale factor, i.e., the position of the cameras is determined up to a metric transformation.

2.4.1 Algorithm

There are several strategies for solving this multiple camera calibration problem. The superior method is bundle adjustment. The process of bundle adjustment is an estimation involving the minimization of the reprojection error, which is defined as the – summed squared – distance between the theoretical image positions of the projections of the estimated 3D points $\mathbf{X}_i$ and the measured image points. Bundle adjustment can handle missing correspondences, which appear if only a subset of cameras shares a common view. However, it involves a nonlinear optimization process and it does not have a direct solution. A sufficiently good starting point is required. We use a hierarchical method to obtain an initial guess for all camera matrices $\mathbf{P_n}^j$ and 3D points $\mathbf{X}_i$. The method is mainly based on the approach presented in [6]. It partitions the set of cameras into manageable subgroups that share a common view. A coordinate

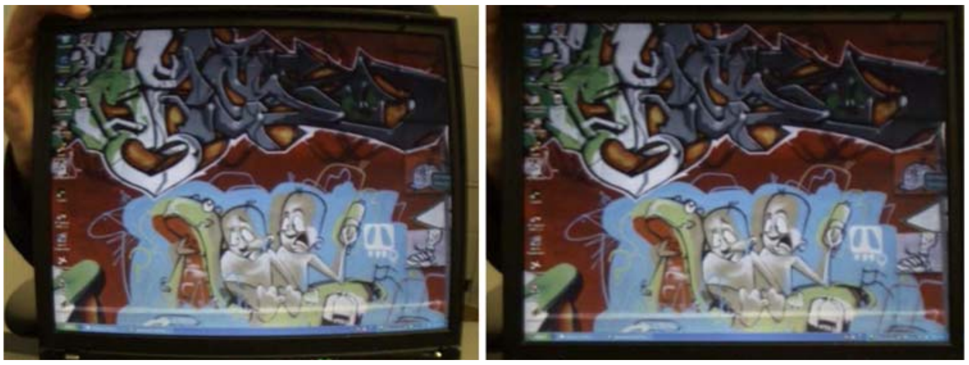


Fig. 7 Original and rectified image

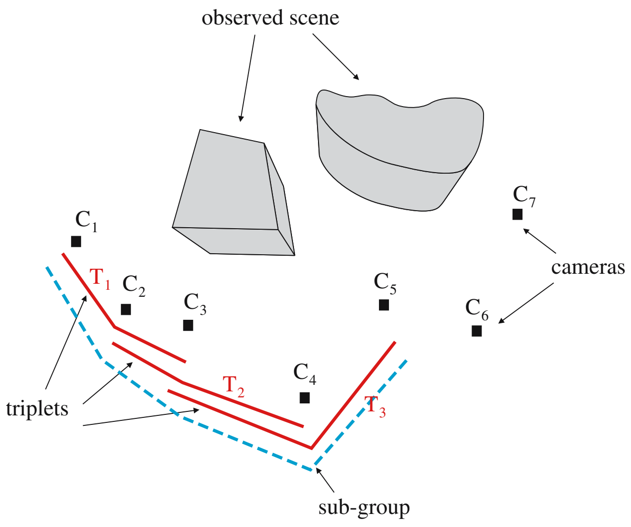


Fig. 8 Camera positions and the structure of the scene are computed by registering the basic building block (triplets of cameras) hierarchically

system is build for each of these subgroups. Based on points and cameras being common to different subsets, these different coordinate frames are merged in a hierarchical fashion in order to build a global coordinate system from the individual overlapping systems. The main advantage contributed by a hierarchical procedure is that the error can be distributed evenly over the entire set of estimated camera matrices. As in [6], we also use image triplets as the basic building block. The cameras' positions in such a basic unit and the structure of the scene observed by three cameras can be computed automatically by calculating the associated trifocal tensor from point correspondences across the three views. Then the triplets are registered into sub-groups, followed by merging these subsets and thus building the entire group. This situation is illustrated in Fig. 8.

The first step in our algorithm is to segment the set of cameras into appropriate subgroups and those subsets into triplets. Consequently neighboring cameras with sufficient view overlaps have to be determined since camera triplets are required to share a common view as a reliable trifocal tensor estimation demandes a sufficient number of point correspondences over the three images. Thus only triplets with a sufficient number of corresponding points are kept. Cameras will only have point correspondences if they are close and share a common view. Thus, even in cases where the set contains many cameras, the number of triplets with a sufficient number of corresponding points will not be large compared to the number of possible camera combinations. Next the triplets need to be clustered into subgroups. Triplets in a certain subgroup and also different subgroups need to share two cameras to enable their registration in a common coordinate frame. Therefore combinations of the above determined triplets are tested. In a second stage the cameras in each triplet are calibrated extrinsically. Robust ways of computing the trifocal tensor and extracting the according camera matrices based on corresponding image points have been extensively studied in [10]. To establish point correspondences, one image of a triple is arbitrarily selected to be the reference image. Two-view point correspondences between this reference image and each of the other two images are then determined by SIFT-feature matching. These correspondences are refined by using them as initializations for the Affine Lucas Kanade feature tracker. The required three-view correspondences are derived by joining the two-view match sets. Features which arise from moving objects can be simply eliminated by removing all keypoints whose positions are temporally unstable. However, the observed 3D scene needs to be sufficiently textured in order to ascertain detection and tracking of enough point features from the acquired images to ensure reliable results in the trifocal tensor estimation.

Registering all triplets into the same coordinate frame is done in a hierarchical manner as proposed in [6]. Registration of triplets and sub-groups is achieved by computing a homography of 3-space between the different metric structures. The objective is to obtain a common set of 3D points and a normalized camera matrix for each view, such that the reprojection error is minimized.

In the following only the registration of two triplets that share exactly two cameras is discussed. All registration problems in the algorithm are analogously solved. In general different overlaps are possible, but as our implementation specifically forces the triplets in a subgroup and the different subgroups to share two cameras, only this case is discussed here. For a detailed description and evaluation of other registration methods the reader is referred to [6]. Given 3D points common in the sets of two different triplets and the homogeneous representation of these points by $\mathbf{X}_i$ in the frame of the first triplet and $\mathbf{X}'_i$ in the second frame (their inhomogeneous representation is denoted with $\bar{\mathbf{X}}_i, \bar{\mathbf{X}}'_i$), their representations in the different metric frames are related by a 3-space homography $\mathbf{H}$ according to

$$\mathbf{X}_i = \mathbf{H}\mathbf{X}'_i \tag{10}$$

Equivalently $\mathbf{P}^j_{\mathbf{n}} = \mathbf{P}'^j_{\mathbf{n}}\mathbf{H}^{-1}$ holds for the corresponding normalized camera matrices of both triplets. The homography between two metric frames can be described by a metric transformation:

$$\mathbf{H} = \begin{pmatrix} \sigma \cdot \mathbf{R} & \mathbf{t} \\ \mathbf{0}_{1\times 3} & 1 \end{pmatrix} \tag{11}$$

where $\mathbf{R}$ denotes a 3×3 rotation matrix and $\mathbf{t}$ a 3×1 translation vector. σ identifies the relative scale between the structures. Since $\mathbf{R}$ can be parametrized by a 3-vector, the transformation between the two different metric frames counts seven unknowns. Two stages are used to derive accurate estimates for those parameters: first a closed-form solution is obtained, which is further refined in a nonlinear stage. In order to compute a direct solution, the first step is to estimate the relative scale σ via the quotient of the mean distances of the 3D points $\bar{\mathbf{X}}_i, \bar{\mathbf{X}}'_i$ to their respective centroid (denoted with inhomogeneous coordinates $\bar{\mathbf{M}}, \bar{\mathbf{M}}'$):

$$\sigma \frac{\frac{1}{n}\sum_{i=1}^{n} \parallel \bar{\mathbf{X}}_i - \bar{\mathbf{M}} \parallel}{\frac{1}{n}\sum_{i=1}^{n} \parallel \bar{\mathbf{X}}'_i - \bar{\mathbf{M}}' \parallel} \tag{12}$$

where $\parallel \cdot \parallel$ denotes the L2-norm and n is the number of common points. Now the second structure is rescaled according to

$$\bar{\mathbf{X}}'_{\mathbf{s}i} = \sigma \bar{\mathbf{X}}'_i \tag{13}$$

so that $\mathbf{X}_i = \mathbf{H}\mathbf{X}'_i$ becomes

$$\mathbf{X}_i = \mathbf{H_s}\mathbf{X}'_{\mathbf{s}i} \tag{14}$$

where

$$\mathbf{H_s} = \begin{pmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}_{1\times 3} & 1 \end{pmatrix} \tag{15}$$

In order to obtain an initial estimate for the coefficients of $\mathbf{R}$ and $\mathbf{t}$ the squared distance between the two structures consisting of points $\mathbf{X}_i$ and $\mathbf{X}'_{\mathbf{s}i}$, is minimized with respect to the coefficients of $\mathbf{H_s}$ using linear algebraic methods:

$$\min_{\mathbf{R},\mathbf{t}} \sum_i d^2(\mathbf{X}_i, \mathbf{H_s}\mathbf{X}'_{\mathbf{s}i}) \tag{16}$$

where $d(\mathbf{x}, \mathbf{y})$ denotes the Euclidean distance between the inhomogeneous points corresponding to $\mathbf{x}$ and $\mathbf{y}$.

Finally the derived initial values are refined in a nonlinear minimization stage where the reprojection error to the originally measured and normalized image points is minimized with respect to all parameters of $\mathbf{H}$:

$$\min_{\sigma,\mathbf{R},\mathbf{t}} \sum_{ij} d^2(\mathbf{P_n}^j\mathbf{H}\mathbf{X}'_i, \mathbf{x}^j_{\mathbf{n}i}) + d^2(\mathbf{P}'^j_{\mathbf{n}}\mathbf{H}^{-1}\mathbf{X}_i, \mathbf{x}^j_{\mathbf{n}i}) \tag{17}$$

This nonlinear minimization is solved using the Levenberg–Marquardt algorithm.

By registering all triplets hierarchically in one common coordinate frame as described above an initial guess for the observed 3D structure (represented by 3D points) and all normalized camera matrices in the entire set of cameras is obtained. Finally a Maximum Likelihood estimate of the entire set of camera positions and the 3D-structure is computed via bundle adjustment:

$$\min_{\mathbf{P_n}^j,\mathbf{X}_i} \sum_{ij} d^2(\mathbf{P}^j_{\mathbf{n}}\mathbf{X}_i, \mathbf{x}^j_{\mathbf{n}i}) \tag{18}$$

Each normalized camera matrix is parameterized by six entries, three representing the rotation matrix and three representing the translation vector. The dimension of the minimization problem adds then up to a total number of $6(N-1)$ parameters for the camera matrices, plus a set of $3L$ parameters for the coordinates of the L reconstructed 3D points.

2.4.2 Experimental results

The extrinsic camera calibration algorithm has been implemented for the case of 11 cameras; the size of the sub-groups was chosen to five cameras. We used cheap web cams for our experiments. Figures 9 and 10 show some of the images taken from the different viewpoints of the cameras in two different experiment. The change of viewpoint between the different locations of the cameras is restriced due to the matching algorithm. The feature extraction algorithm requires the scene to

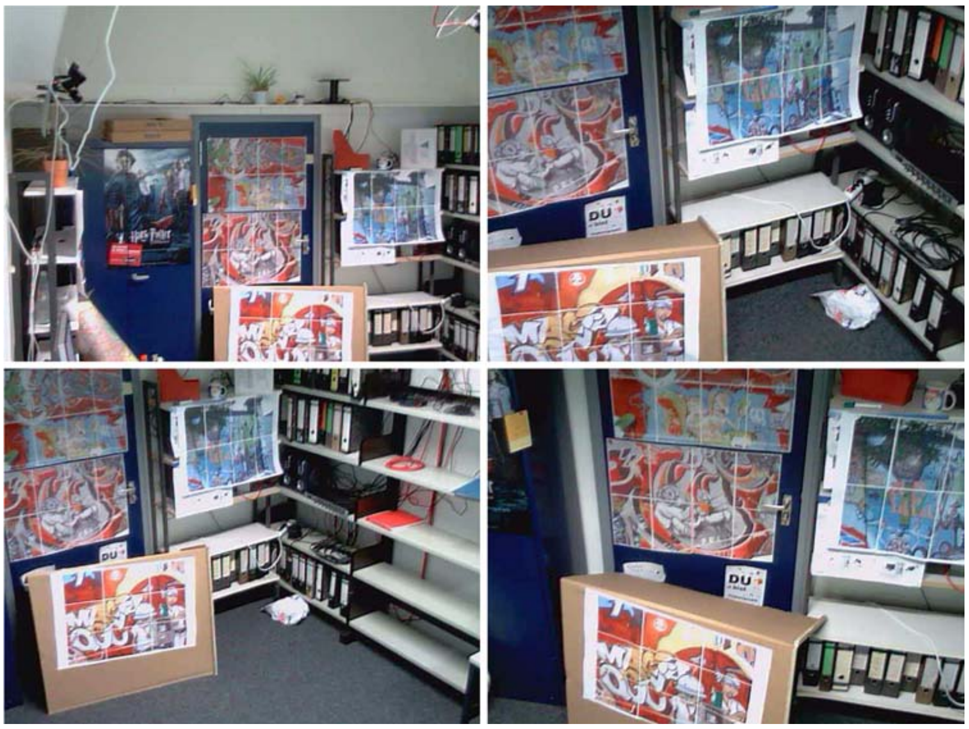


Fig. 9 Example images of the lab scene from different viewpoints (cameras)

be sufficiently texured. Sub-pixel matching was required to obtain accurate results in both experiments. We also conducted experiments where we used affine invariant feature detectors [16] instead of the SIFT detector, but those did not improve our results.

The resulting camera positions and scene reconstructions are shown in Figs. 11 and 12. Camera positions are marked with yellow pyramids, reconstructed scene points with blue dots. In Fig. 13 the final reprojection error is illustrated for one estimated camera in our lab scene experiment. The distance between the reprojected points and the measured image points is very small. Therefore the overall estimation is highly accurate.

Discussion: In the given examples the implementation performs very well. However experiments with different data sets have shown that sporadically the accuracy of the algorithm can be severly affected. Thorough analysis showed that mis-estimations were caused by inaccurately estimated triplets. If the camera positions and/or the 3D points in one triplet are estimated inaccurately, the homography estimation to register this triplet in a subgroup fails as well. As a result the whole subgroup configuration is determined incorrectly leading to an initial guess for the entire group too far away from the actual value. As the optimization problem of the final bundle adjustment is of very high dimension, a poor initial guess commonly results in the nonlinear optimization to fail completely, i.e., to converge to a suboptimal solution or to not converge at all.

One source of failure in the triplet estimation may consist in corresponding image points that are not extracted sufficiently accurate, due to the performance limits of the feature extraction and matching algorithm and/or the feature tracker. Those algorithms are only partly invariant to perspective transformations. Another cause of failure arises from the fact that the intrinsic camera parameters can only be estimated with a certain accuracy. This may also have an impact on the noise level in the corresponding normalized points.

3 Optimizing coverage

In the previous section we have shown how to calibrate all visual sensors. Now we focus on post-optimizing their poses to increase coverage. We assume, that the camera positions in a given space are known. As the calibration procedure presented in this paper only enables us to determine the relative positions of the cameras up to a global scale factor, we need to determine this global scale factor. This can be done e.g., by placing a known calibration pattern in the scene, visible by at least two cameras. As the absolute distance of two points on this pattern is known, the absolute distance of the two cameras observing the pattern can be calculated and thus the overall scale factor is determined. Another option is to place microphones and loudspeakers close to the

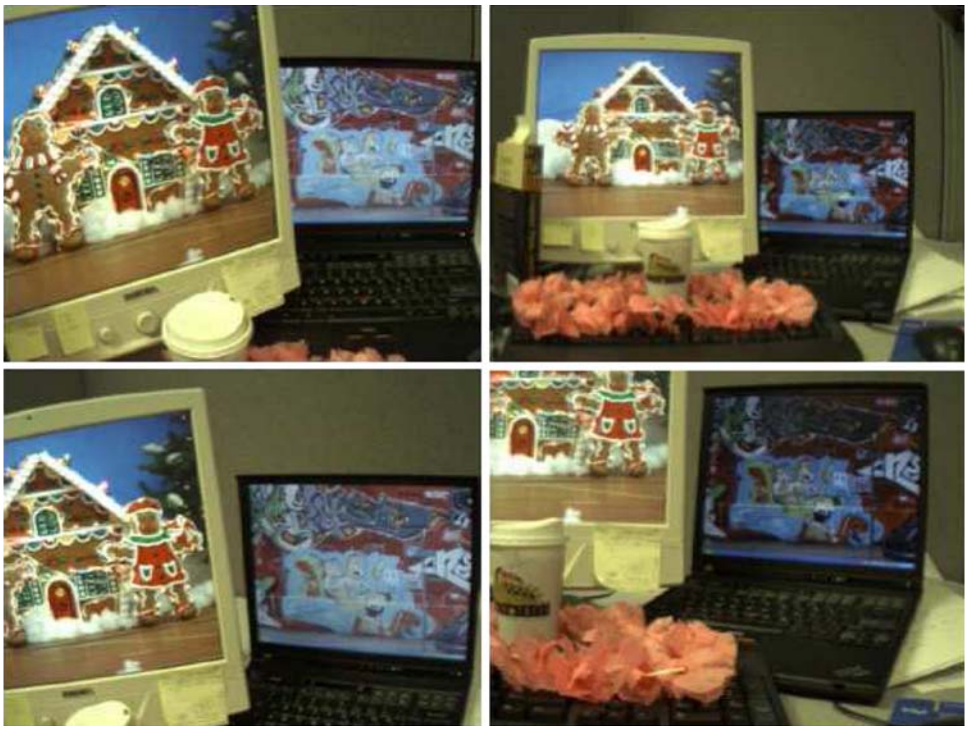


Fig. 10 Example images of an office desk scene from different viewpoints (cameras)

cameras and to determine the absolute distance by the algorithm presented in [21].

Starting from this knowledge our aim is to increase the coverage of the space observed by calculating new optimal pan and tilt angles for each camera such that coverage of the space at a given sampling frequency is maximized. This optimization can be done with any type of camera. However, with pan-tilt cameras we can remotely and automatically drive the cameras into the right poses. This allows for an interative optimization approach that can easily handle inaccuracies in the estimation and control part.

The new poses of the cameras have to be calibrated again, as the automatic pan-tilt navigation is often imperfect. This could be done as described in Sect. 2 or by recording images during the cameras' motion and then perform pose and position calibration, i.e., extrinsic calibration, with those images similar to the method presented in Sect. 2.

3.1 Problem statement

Definitions: In the following the term *space* denotes a convex physical three-dimensional room.

A point in that space is *covered* if that point is captured with a required minimal resolution. The minimal resolution is satisfied if the point in space is imaged by at least one pixel of a camera that does not aggregate more than $x\ cm^2$ of a surface parallel to the imaging plane through that point. x is expressed in terms of the sampling frequency f_s and converted into the field-of-view of a camera. The *field-of-view* of a camera is defined as the volume in which a pixel aggregates no more than $\frac{1}{f_s^2} cm^2$ of a surface parallel to the imaging plane. Thus an object that appears in the camera's field-of-view is imaged with at least this resolution assuming the object has a planar surface orthogonal to the optical axis.[1] Occlusions are not considered.

Problem statement: Given a space to be covered at a sampling frequency f_s by visual sensors, we are interested in the following problem: given a set of cameras and their current positions and initial poses in space, determine their new poses, defined by means of pan and tilt angles, such that coverage is maximized.

3.1.1 *Modeling a camera's field-of-view*

We consider a simple model for our cameras. Since each camera is assumed to be able to pan and tilt, the possible camera motion is modeled as two idealized rotations around the origin. This simple pan-tilt motion model is shown in Fig. 14 a. A rotation around the x-axis and then around the y-axis correspond to tilt and pan, respectively. Our simple model assumes that the pan and tilt axes are orthogonal, aligned with the image plane, and through the cameras optical center. The field-of-view

[1] Clearly the resolution is smaller if the surface is not orthogonal.

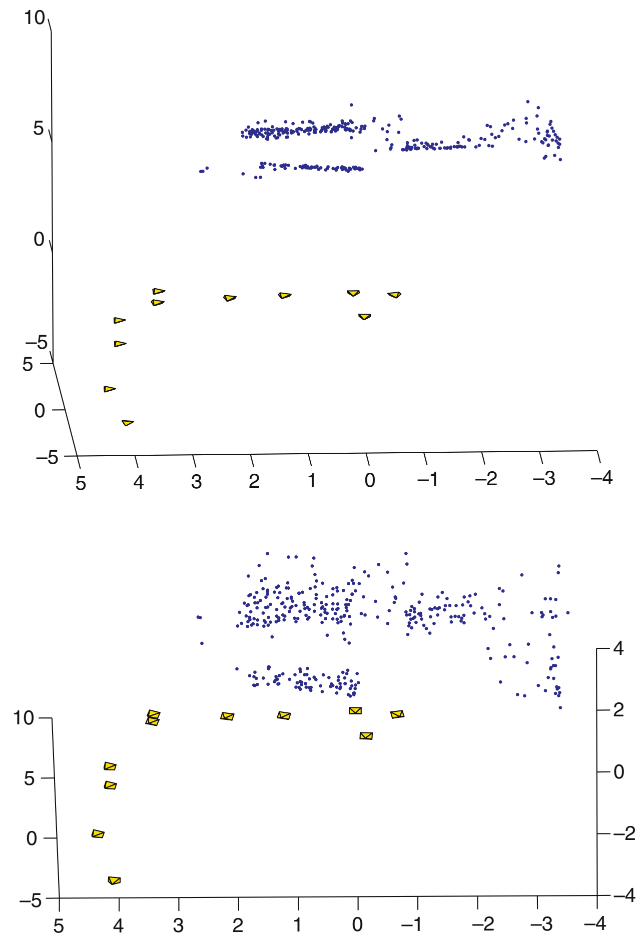


Fig. 11 Two different views of the reconstructed 3D scene points and camera positions for the entire group of 11 cameras for the lab scene experiment

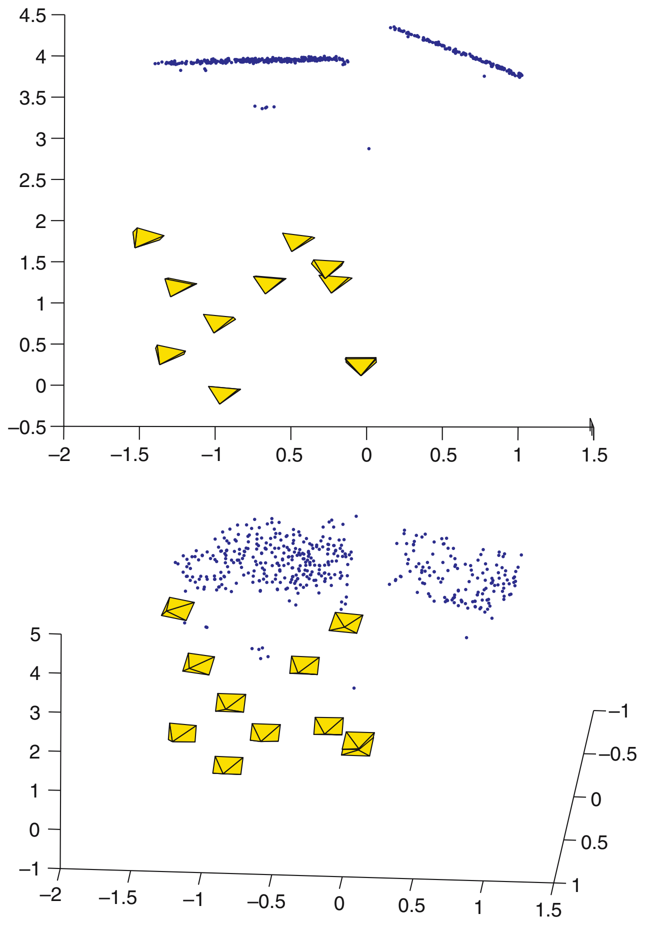


Fig. 12 Two different views of the reconstructed 3D scene points and camera positions for the entire group of 11 cameras for the office desk scene

can then be described by a pyramid. The parameters of this pyramid can be easily calculated given the intrinsic camera parameters and the sampling frequency f_s using well known geometric relations.

Defining the field-of-view by a pyramid enables us to describe the area covered by a camera at position (c_x, c_y, c_z) in the world coordinate system and pose $(\mathbf{R}, \alpha, \beta)$ linearly. $\mathbf{R}$ denotes here the inital pose of the camera, α and β denote the variable tilt and pan angles that are varied to optimize the coverage. It should be mentioned that bold letter denotes vectors such as $\mathbf{X}$ and x the first component of that vector.

To define our field-of-view, we express the coordinate vectors of points in the final camera coordinate system C as a function of the coordinate vectors of the same points in the world coordinate system A, i.e., we transform points from the world coordinate system to the camera coordinate system (see Fig. 14 b). Therefore we first transform the point to the inital camera coordinate system B (without pan and tilt) and from there to C, the final camera coordinate system including pan and tilt angle. We denote by ${}^{F}\mathbf{X}$ the coordinate vector of the point $\mathbf{X}$ in the frame F. According to [7], the transformation of a point from a coordinate system A to another coordinate system B is expressed by :

$$ {}^{B}\mathbf{X} = {}^{B}_{A}\mathbf{R}\,{}^{A}\mathbf{X} + {}^{B}\mathbf{O}_A \tag{19}$$

where ${}^{B}\mathbf{O}_A$ denotes the origin of the world coordinate system A in coordinate system B, i.e.,:

$$ {}^{B}\mathbf{O}_A = -\mathbf{t} = \begin{pmatrix} -c_x \\ -c_y \\ -c_z \end{pmatrix} \tag{20}$$

The frames B and C are seperated by pure rotation. This rotation models the motion of a pan-tilt camera. The coordinate system attached to the cameras origin rotates around the x- and y-axis, by α and β respectively:

$$ {}^{C}\mathbf{X} = {}^{C}_{B}\,\mathbf{R}\,{}^{B}\mathbf{X} \tag{21}$$

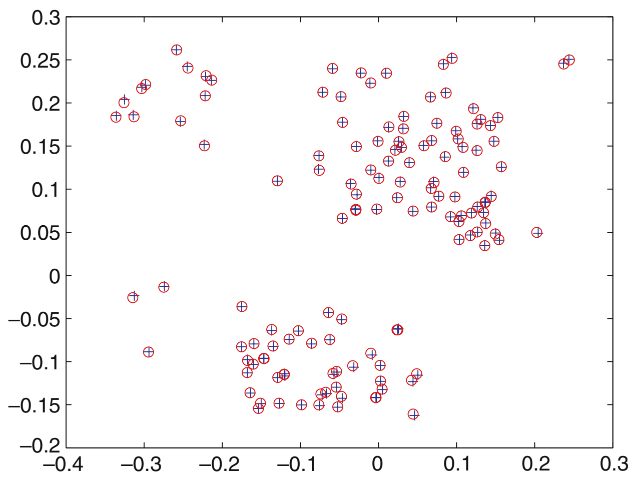


Fig. 13 Final reprojection error illustrated in one camera image

where

$$ {}^{C}_{B}\mathbf{R} = \mathbf{R_y}\mathbf{R_x} \tag{22}$$

and

$$\mathbf{R_x} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha) & \sin(\alpha) \\ 0 & -\sin(\alpha) & \cos(\alpha) \end{pmatrix} \tag{23}$$

$$\mathbf{R_y} = \begin{pmatrix} \cos(\beta) & 0 & -\sin(\beta) \\ 0 & 1 & 0 \\ \sin(\beta) & 0 & \cos(\beta) \end{pmatrix} \tag{24}$$

Concatenating the transformation gives the coordinates of a point in the world coordinate system transformed to its coordintates in the final camera coordinate system:

$$ {}^{C}\mathbf{X} = {}^{C}_{B}\,\mathbf{R}^{B}_{A}\mathbf{R}^{A}\mathbf{X} - {}^{C}_{B}\,\mathbf{R}\cdot\mathbf{t} \tag{25}$$

The resulting area covered by the field-of-view of a certain camera can now be defined by five equations (see Fig. 14 c):

$$z_c \leq d \tag{26}$$

$$y_c \leq \frac{a}{2d}\cdot z_c \tag{27}$$

$$y_c \geq -\frac{a}{2d}\cdot z_c \tag{28}$$

$$x_c \leq \frac{b}{2d}\cdot z_c \tag{29}$$

$$x_c \geq -\frac{b}{2d}\cdot z_c \tag{30}$$

3.1.2 Modeling space

We only consider convex spaces without obstacles constricting the field-of-view of our visual sensors. In order to define coverage, we sample the space by means of regular grid points. The minimum distance Δ between two grid points in the x-, y- and z- direction is determined by the *spatial sampling frequency* f_a: $\Delta = 1/f_a$. With that our problem turns into a grid coverage problem. In order to optimize coverage, we determine the camera poses that cover the largest percentage of grid points in the space. For $f_a \to \infty$ our approximated solution converges to the continuous-case solution.

If some parts of the room are known to be more important, e.g., at doors, a higher weighting can be given to those parts by sampling here with a higher frequency, whereas e.g., parts that are less likely interesting might be sampled with a lower frequency.

In the ideal case cameras' poses are continuously in the space, i.e., the variables α and β that define a camera's pan and tilt are continuous variables. As we are not able to solve our problem for the continuous case we approximate the continuous case by sampling the poses. Cameras can only adopt those discrete poses.

3.2 Linear programming

Considering N cameras that are calibrated, i.e., their fields-of-view as well as positions in the space are known, we formulate our camera positioning problem in terms of maximizing the coverage. We assume for notational convenience, that our space consists of s_x,s_y and s_z grid points in the x- and y- and z-dimension respectively.[2] Similarly we discretize the angles α and β defining a camera's tilt and pan to s_α and s_β different angles only. A camera at position (c_x, c_y, c_z) with orientation $\mathbf{R}$ and pan and tilt α and β respectively covers a grid point (x, y, z) if and only if (26)–(30) are satisfied.

Thus, we can state the optimization problem as follows: given a set of grid points and camera models, maximize the coverage by optimally assigning pan and tilt angles to cameras.

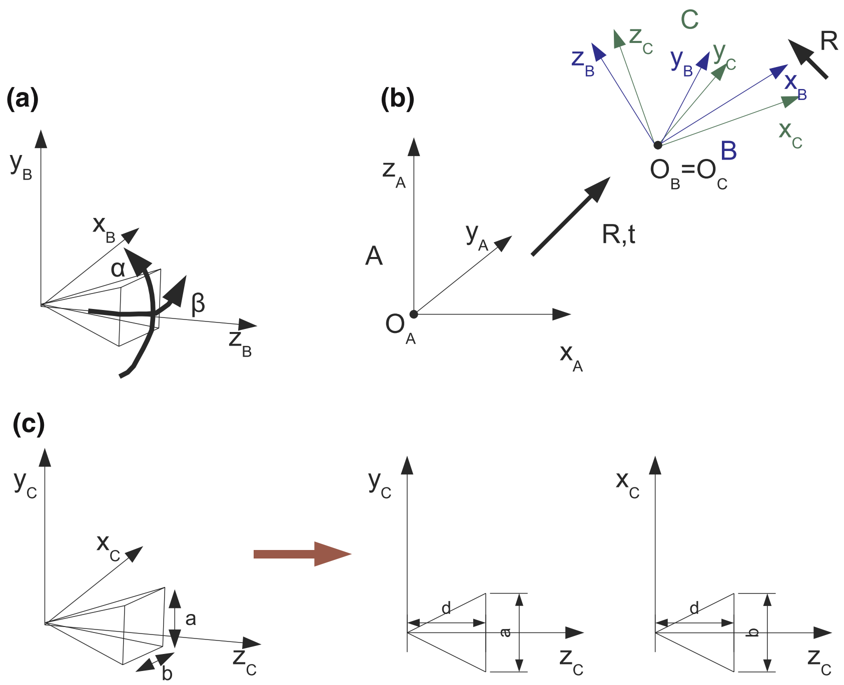
In the following we derive an binary integer programming (BIP) model to solve this problem. Our approach is based on the algorithm presented in [3]. First we define some binary variables. Let a binary variable c_{ijk} be defined by:

$$c_{ijk} = \begin{cases} 1 & \text{if grid point } (i,j,k) \text{ is covered by a} \\ & \text{minimum of } M \text{ cameras} \\ 0 & \text{otherwise} \end{cases} \tag{31}$$

where $M \geq 1$ denotes the minimum number of cameras that should cover each grid point. This variable will be usually chosen to 1, but can be easily changed to e.g., two cameras or more. This is e.g., usefull for the case that an object at a certain grid point should be exactly located

[2] Given a rectangular space s_x, s_y and s_z can be easily calculated given the room's dimensions and the spatial sampling rate f_a.

Fig. 14 Deriving the model of a camera's field-of-view



in depth. This is only possible if the object is viewed by at least two cameras. Then the position of the object can be calculated by triangulation.

The total number Nb of covered sample points is then given by

$$Nb = \sum_{i=0}^{s_x-1} \sum_{j=0}^{s_y-1} \sum_{k=0}^{s_z-1} c_{ijk} \tag{32}$$

We define two further binary variables $x_{n\alpha\beta}$ and $g^n(\alpha, \beta, i, j, k)$:

$$x_{n\alpha\beta} = \begin{cases} 1 & \text{if camera } n \text{ at point } (c_x, c_y, c_z) \text{ with} \\ & \text{initial orientation } \mathbf{R} \text{ has tilt } \alpha \text{ and pan } \beta \\ 0 & \text{otherwise} \end{cases} \tag{33}$$

$$g^n(\alpha, \beta, i, j, k) = \begin{cases} 1 & \text{if a camera at point } (c_x, c_y, c_z) \\ & \text{with intial orientation } \mathbf{R} \\ & \text{and tilt } \alpha \text{ and pan } \beta\text{, covers} \\ & \text{grid point } (i, j, k) \\ 0 & \text{otherwise} \end{cases} \tag{34}$$

$g^n(\alpha, \beta, i, j, k)$ can be calculated in advance for each camera and stored in a table.

We now need to express the variables that define coverage in terms of the other above defined variables. This is done as shown below. Since $c_{ijk} = 1$, if and only if at least M cameras cover the grid point (i, j, k) we introduce the following inequality for each grid point:

$$c_{ijk} \cdot \left(\sum_{n,\alpha,\beta} x_{n\alpha\beta} \cdot g^n(\alpha, \beta, i, j, k) - M \right) \geq 0 \tag{35}$$

The constraint (35) involves a product of binary variables, thus it is nonlinear. In order to linearize the inequality, we introduce a new binary variable for the appearance of this nonlinear term, as well as two additional constraints [29]. Therefore we replace $c_{ijk} \cdot x_{n\alpha\beta}$ by the binary variable $v_{ijkn\alpha\beta}$ and introduce the following constraints:

$$c_{ijk} + x_{n\alpha\beta} \geq 2 \cdot v_{ijkn\alpha\beta} \tag{36}$$

$$c_{ijk} + x_{n\alpha\beta} - 1 \leq v_{ijkn\alpha\beta} \tag{37}$$

To ensure that exactly one pan-tilt combination is assigned to each camera, we also need for each camera the constraint:

$$\sum_{\alpha,\beta} x_{n\alpha\beta} = 1 \tag{38}$$

Our sensor deployment problem can now be formulated as an BIP model:

$$\max \sum_{i=0}^{s_x-1} \sum_{j=0}^{s_y-1} \sum_{k=0}^{s_z-1} c_{ijk} \tag{39}$$

subject to the constraints:

$$c_{ijk} + x_{n\alpha\beta} \geq 2 \cdot v_{ijkn\alpha\beta} \tag{40}$$

$$c_{ijk} + x_{n\alpha\beta} - 1 \leq v_{ijkn\alpha\beta} \tag{41}$$

$$\text{for} \quad 0 \leq i \leq (s_x - 1), \quad 0 \leq j \leq (s_y - 1), \quad 0 \leq k \leq (s_z - 1), \quad 1 \leq n \leq N, \quad 0 \leq \alpha \leq (s_\alpha - 1), \quad 0 \leq \beta \leq (s_\beta - 1)$$

$$\sum_{n,\alpha,\beta} v_{ijkn\alpha\beta} \cdot g^n(\alpha, \beta, i, j, k) - c_{ijk} \cdot M) \geq 0 \tag{42}$$

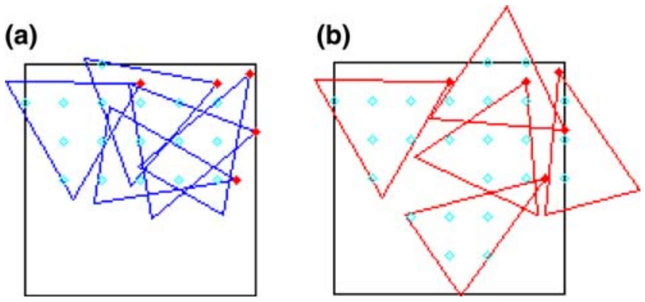


Fig. 15 Results obtained for the 2D case: **a** start configuration (coverage=34.69%); **b** optimized configuration (coverage=49.98%)

for $0 \le i \le (s_x - 1), \quad 0 \le j \le (s_y - 1), \quad 0 \le k \le (s_z - 1)$

$$\sum_{\alpha,\beta} x_{n\alpha\beta} = 1 \tag{43}$$

for $1 \le n \le N$

$$c_{ijk}, v_{ijkn\alpha\beta}, x_{n\alpha\beta} \in \{0, 1\} \tag{44}$$

for $0 \le i \le (s_x - 1), \quad 0 \le j \le (s_y - 1), \quad 0 \le k \le (s_z - 1),$
$1 \le n \le N, \quad 0 \le \alpha \le (s_\alpha - 1), \quad 0 \le \beta \le (s_\beta - 1)$

The number of variables and constraints depends on the number of grid points and samples. Thus, if we increase the number of grid points and possible pan-tilt configurations to achieve a better approximation of the continuous case, the number of variables and constraints in our BIP model increases accordingly.

3.3 Experimental results

The above presented linear programming approach has been implemented in C++ using the *lpsolve* package [1]. However for better visualization we first present results obtained in the 2D case and subsequently we report our results for 3D.

Figure 15 illustrates a result for one configuration obtained by solving our implemented BIP model in the 2D case. On the left side, the initial configuration is shown, while on the right side the optimized configuration is depicted. Red dots mark camera positions, blue and red triangles illustrate the cameras' field-of-views. Light-blue dots mark covered grid points. The coverage of the space increases in this experiment from 34.69 to 49.98%. We restricted the pan and tilt angle α and β to be sampled in the range of $\pm 45°$, as this seems a realistic range for pan-tilt cameras.

Table 2 shows results obtained with our optimization algorithm in 3D for different parameter settings. The number of cameras has been set to three in all experiments. The x- and y-dimension of the considered space are set to 4 units, the z-dimension to 2 units. The intial configuration of the cameras is shown in Fig. 16. Yellow pyramids mark the initial cameras' position and orientation, red pyramids mark the optimized cameras' poses for $f_a = 0.8, s_{\alpha/\beta} = 5$. For better visualization the cameras' field-of-views are not shown. In our optimization we restriced the pan and tilt angles to be sampled in the range of $\pm 45°$. The results clearly demonstrate the gain in coverage in every experiment.

Table 2 Results obtained for different parameter settings by solving our BIP model in 3D

N	3	3	3	3
f_a	0.5	0.5	0.8	1.0
$s_{\alpha/\beta}$	5	7	5	3
percentage of space covered before optimization	33.33	33.33	37.50	34.67
percentage of space covered after optimization	55.56	61.11	65.69	52.00

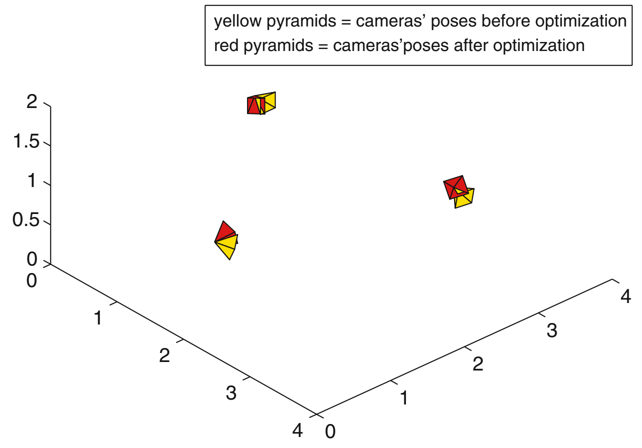


Fig. 16 Example result obtained by solving our BIP model in 3D

The above presented BIP problem is practically solvable for only a small number of grid points. For a large number appropriate solutions need to be developed.

4 Conclusion

In this paper we have presented a flexible and easy way to calibrate multiple cameras in a distributed platform of GPCs. Our method needs minimal user intervention. Hence the proposed method can be used in a variety of places ranging from single desktop cameras to multi-camera lab setups. All stages of the calibration algorithms have been implemented and experimental results on real data showed that the presented methods work very well. As the change in viewpoint between the different cameras is restricted, future work is needed to improve the automatic extraction of point correspondences between images.

We have also derived an LP approach for post-optimizing the camera poses to increase coverage of the space observed. Our experimental results demonstrate the gain in coverage. Future work on this topic will include the investigation of how to handle large numbers of grid points.

Acknowledgements Special thanks to Jean-Yves Bouguet and Walter Kellermann for their valuable advise and support on the calibration work.

References

1. Berkelaar, P.N.M., Eikland, K.: lpsolve: Open souce (mixed-integer) linear programming system. Eindhoven U. of Technology. http://www.groups.yahoo.com/group/lp_solve/files/Version5.5/
2. Bouguet, J.-Y.: Camera Calibration Toolbox for Matlab. http://www.vision.caltech.edu/bouguet/calib_doc/
3. Chakrabarty, H.Q.K., Iyengar, S.S., Cho, E.: Grid coverage for surveillance and target location in distributed sensor networks. IEEE Trans. Comput. **51**(12), 1448–1453 (2002)
4. Chen, X., Davis, J., Slusallek, P.: Wide area camera calibration using virtual calibration objects. In: Proceedings of CVPR '00, pp. 2520–2527 2000
5. Erdem, U., Sclaroff, S.: Optimal placement of cameras in floorplans to satisfy task requirements and cost constraints. In: OMNIVIS Workshop, 2004
6. Fitzgibbon, A., Zissermann, A.: Automatic camera recovery for closed or open image sequences. In: Proceedings of European Conference on Computer Vision, pp. 311–326 1998
7. Forsyth, D.A., Ponce, J.: Computer Vision: A Modern Approach. Prentice Hall Professional Technical Reference, 2002
8. Harris, C., Stephens, M.: A combined corner and edge detector. In: Proceedings of the 4th Alvey Vision Conference, pp. 147–152, 1988
9. Hartley, R.: An algorithm for self-calibration from several views. In: Proceedings of CVPR '94, pp. 908–912, Seattle, USA, 1994
10. Hartley, R., Zisserman, A.: Multiple View Geometry in Computer Vision. Cambridge University. Press, Cambridge (2003)
11. Heikkilä, J., Silven, O.: A four-step camera calibration procedure with implicit image correction. In: Proceedings of CVPR '97, pp. 1106–1112, 1997
12. Hörster, E., Lienhart, R.: Approximating optimal visual sensor placement. In: Proceedings of ICME '06, 2006
13. Intel corporation. OpenCV Computer Vision Library. http://www.intel.com/research/mrl/research/opencv/
14. Lienhart, R., Kozintsev, I., Wehr, S., Yeung, M.: On the importance of exact synchronization for distributed audio processsing. In: Proceedings of ICASSP '03, pp. 840–843, 2003
15. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. Int. J. Comput. Vis, **60**(2), 91–110 (2004)
16. Mikolajczyk, K., Schmid, C.: A performance evaluation of local descriptors. In: Proceedings of CVPR '03, vol. 2, pp. 257–263, 2003
17. Mikolajczyk, K., Schmid, C.: An affine invariant interest point detector. In: ECCV (1), pp. 128–142, 2002
18. Mittal, A., Davis, L.: Visibility analysis and sensor planning in dynamic environments, vol. I, pp. 175–189, 2004
19. Pollefeys, M.: Self-Calibration and Metric 3D Reconstruction from Uncalibrated Image Sequences. PhD thesis, K. U. Leuven, Belgium (1999)
20. Rahimi, A., Dunagan, B., Darrell, T.: Simultaneous calibration and tracking with a network of non-overlapping sensors. CVPR **01**:187–194 (2004)
21. Raykar, V., Kozintsev, I., Lienhart R.: Position calibration of audio sensors and actuators in a distributed computing platform. In: Proceedings ACM Multimedia '03, pp. 572–581, 2003
22. Sahni, S., Xu, X.: Algorithms for wireless sensor networks. Int. J. Distrib. Sensor Netw. **1**(1), 35–56 (2005)
23. Shi, J., Tomasi, C.: Good features to track. In: Proceedings of CVPR '94, pp. 593 – 600, 1994
24. Sinha, S.N., Pollefeys, M.: Calibrating a network of cameras from live or archived video. In: Proceedings of CVPR '04, 2004
25. Sturm, P., Triggs, W.: A factorization based algorithm for multiple-image projective structure and motion. In: Proceedings of European Conference on Computer Vision, pp. 709–720, 1996
26. Svoboda, T., Martinec, D., Pajdla, T.: A convenient multi-camera self-calibration for virtual environments. PRESENCE Teleoper. Virtual Environ. **14**(4), 407–422 (2005)
27. Tsai, R.: A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf tv cameras and lense. IEEE J. Rob. Autom. **RA-3**, 323–344 (1987)
28. Wang, J., Zhong, N.: Efficient point coverage in wireless sensor networks. J. Comb. Optim. **11**(3), 291–304 (2006)
29. Williams, H.: Model Building in Mathematical Programming 2nd edn, Wiley, New York, (1985)
30. Zhang, Z.: A flexible new technique for camera calibration. Technical Report MSR-TR-98-71, Microsoft Research, Redmond, USA, 1998
31. Zou, Y., Chakrabarty, K.: Sensor deployment and target localization in distributed sensor networks. Trans. Embed Comput Syst. **3**(1), 61–91 (2004)