Interactive Building and Augmentation of Piecewise Planar Environments Using the Intersection Lines

Gilles Simon, Marie-Odile Berger

The Visual Computer manuscript No.
(will be inserted by the editor)

Interactive Building and Augmentation of Piecewise Planar Environments Using the Intersection Lines

Gilles Simon · Marie-Odile Berger

Received: date / Accepted: date

Abstract This paper describes a method for online interactive building of piecewise planar environments for immediate use in augmented reality. This system combines user interaction from a camera-mouse and automated tracking / reconstruction methods to recover planar structures of the scene that are relevant for the augmentation task. An important contribution of our algorithm is that the process of tracking and reconstructing planar structures is decomposed into three steps - tracking, computation of the intersection lines of the planes, reconstruction- that can each be visually assessed by the user, making the interactive modeling procedure really robust and accurate with intuitive interaction. Videos illustrating our system both on synthetic and long real-size experiments are available at
`http://www.loria.fr/~gsimon/vc`.

Keywords Interactive building · Structure-from-motion · SLAM · Particle filtering · Camera tracking · Augmented reality

1 Introduction

Augmented Reality (AR) has now progressed to the point where real-time applications are being considered and needed. Computer vision techniques greatly contributed in achieving the required reliability and accuracy of the positioning systems. Marker-based techniques [16] as well as those based on a CAD model of parts of the observed scene [20] have been used successfully in many areas. The scene (or marker) measurements are used both to compute the camera pose and define the position and orientation of the 3-D virtual objects with regard to the real world. In this paper, we want to go one step beyond by being able to perform AR in *a priori* unknown environments. More precisely, we aim to track a calibrated hand-held camera in a previously unknown scene without any known initialization target, while building a 3-D map of this environment and populating this map with virtual objects. This may be of particular interest in some collaborative scenarios where users add and share information in an environment they are discovering together [15]. It would also enable in-situ prototyping e.g. of home landscape designs or visual special effects during the footing stage.

Past years have seen the emergence of simultaneous localization and mapping (SLAM) vision-based techniques for estimating the pose of a moving monocular camera in unknown environments [13,4,5,17,19]. However, only few of these works address the problem of adding virtual objects into the map while it is being built. A minimum requirement to be able to perform this task is that some planar surfaces are identified into the map upon which the virtual objects can be placed with correct orientation. Planar surfaces also make easy to handle self-occlusions of the map as well as collisions, occlusions, shadowing and reflections between the map and the added objects. Unfortunately, automatic detection of multiplanar surfaces is far from been safe, as it is discussed in section 2. Most systems use a number of thresholds and these have a noticeable effect on system performance. Moreover, because these techniques produce an unsorted point-cloud or mesh model, they lack the ability to describe the scene in terms of separable well defined objects suitable for use in AR systems. The benefit of using semi-automatic, rather than fully automatic algorithms, is that we can model relevant structures for augmentation.

G. Simon
LORIA, Nancy University
Campus Scientifique, BP 239,
54506 Vandœuvre-les-Nancy, FRANCE
Tel.: +33-3-83-59-20-67
E-mail: gsimon@loria.fr

M.-O. Berger
LORIA, INRIA Nancy-Grand Est

Some interactive systems have been designed to online modeling of scenes [8,2,30]. For instance, [2] enables the definition of 3-D models through a series of user input actions and 3-D gestures. Object's vertices are clicked in one frame using the "camera-mouse" principle (see below). This provides a 3-D ray of possible positions of the vertex in space. The epipolar line is then computed for every frame as the camera is moved to a different viewpoint, and the user has to scroll with the wheel the current estimate of the vertex's depth along the line until the vertex's projection is aligned with the true object's vertex in the video. After creating two vertices, a 3-D line can be defined, and after creating three vertices, a 3-D plane can be defined. It is also possible to use an existing plane to constrain new vertices, and faces can be extruded into volumes using the wheel.

In this paper, our goal is not to have a complete or overly detailed geometric scene model. We only need to consider those parts of the scene that are relevant to camera tracking and virtual object positioning. We thus propose a simpler and faster procedure to define the environment, based on paintbrush-like drawing in the video stream and automatic method to recover plane equations from areas outlined by the user.

2 Further Related Works

The main purpose of this section is to show why interactivity is needed for plane discovery in AR applications. [3] presents a simple AR game in which an agent has to navigate using real planar surfaces in a scene. The RANSAC algorithm [7] is used at each frame of operation of a SLAM system to search for planes in the 3-D point-cloud map. Plane hypotheses are generated from minimal sets of points randomly sampled from the subset of point features with sufficient confidence. A point is deemed to be in consensus with the hypothesis if its perpendicular distance to the plane, d, is less than a suitably chosen threshold d_T, e.g. $d_T = 0.5cm$. As well as looking for new planes in each frame, the system also considers new 3-D points added to the SLAM map as candidates for addition to existing planes. A 3-D point is added to an existing plane if $d < d_T$. An obvious drawback of this method is that tuning the d_T threshold may be difficult in practice as this value depends on the size of the scene. Moreover, this requires that the map is scaled using some physical measurements of the scene. In [3], a marker whose size is known is used to initialize the map, but markers are not easily usable on a larger-than-room scale. Several other limitations of this approach are also pointed out by the authors in the conclusion of [9].

The scale-dependent threshold problem may be tackled by segmenting planes into the 2-D video images instead of the 3-D point-cloud. A lot of works have been devoted these last years to identifying multiple homographies between an image pair where point correspondences have been established. The so-called "Sequential RANSAC" has been proposed as a solution (e.g. in [32]): this algorithm consists in iteratively applying RANSAC on the set of correspondences, from which detected inlier groups are withdrawn after each iteration. However, as pointed out by various authors [35,28,23], this approach suffers from strong limitations [23]: detection of false homographies (validation of groups that are composed of outliers), fusion of nearby homographies (two or more homographies are detected as the same consensus set), segmentation of the consensus set of a single homography into smaller ones (e.g. when spatial tolerance is too small), and so on. To tackle these issues, [35] introduced the multi-RANSAC algorithm. The strategy is to detect all homographies simultaneously by fusing the different groups found by RANSAC. The method is effective, but the number of homographies to be found is user specified, which is not acceptable in our application context. Other methods have been proposed that do not require specifying the number of homographies [34,28,23]. Nevertheless, their practical use generally still requires the setting of one or several sensitive parameters. Only the *a contrario* approach does not need any parameters at all [22,23], but this approach is highly combinatorial and can not reach real-time requirement.

A common problem shared by all these approaches is their dependence on the feature density and distribution in the image stream. More explicitly, extracting bounded planar surfaces from sets of detected-as-coplanar points can lead to two inverse problems. On the one hand, it can lead to fill the gap in between non-connected surfaces (in [3], this is partly tackled by introducing another distance threshold). On the other hand, it can lead to disconnecting two parts of the same surface, or severely clip a surface e.g. if a localized group of features is at the center of a large uniform surface (see for instance the walls in Fig. 8). Incorporating reconstruction of sparse 3-D line segments and dense photo-consistency in multiple view may help to avoid these problems [26], but at the expense of unacceptable increase in computation time (2 to 3 minutes per image in [26]).

Integrating human user input will allow us to safely segment and reconstruct relevant pieces of planes from a video stream. Adding user input in a SLAM process is quite natural as SLAM processes already imply active manipulation of the camera, if only for getting the required parallax motions. For instance, in [17], user cooperation is used for initializing the map: when the system is started, the user places the camera above the workspace and presses a key to capture a first key-frame. The user then smoothly moves the camera to a slightly offset position and makes a second key-press that provides a second key-frame. Some point features are tracked between the two key-frames from which the base map is triangulated using a five-point stereo algorithm (all

this procedure has been recently implemented on a mobile phone [19]). The main limitation of [17] with regard to our goals is that only one plane, the dominant plane, is detected in the point-cloud map.

This paper is an expanded version of [25]. It provides detailed proofs of corollary 1 and 2, and the detailed explanation for the different stages of the system (sections 5 to 8). It also provides some extra experimental results shown in section 9.4.

3 System Overview

Our system is designed for a scene containing multiple planes. One of these planes, called the *reference plane*, has to be partially visible during all the mapping operations. In standard use of the system, the reference plane is the ground plane and the other planes are walls.

User input is performed using a hand-held camera and four keyboard keys (the directional arrows). The camera-mouse method is used to define the planar regions (called 2-D blobs) in the image stream. This method consists in selecting objects by pointing at them through the camera. A fixed cursor, generally a cross at the center of the camera window, is used for ray-casting selection [2,12]. In our interface, the cursor is not a cross but a circle and we do not have one but two cursors (Fig. 7): one circle on the bottom half of the screen is used to define 2-D blobs on the reference plane and another circle on the top half of the screen is used to define 2-D blobs on the other planes. Pressing the up or down key enables the operator to freeze the related circle into a blob which is immediately tracked in subsequent frames using RANSAC matching of Harris corners[1] inside the blob. Each time the user presses the key again, a convex hull is computed between the tracked blob and the related circle, forming a new blob which at its turn is tracked, and so on, until the blob is fully defined.

Different tasks are performed during a working session that are described below. In order to better understand how things work in practice, the reader is invited to watch the videos associated with this paper [2].

Map initialization. Two blobs are indicated on non-parallel planes using the camera-mouse. These blobs are tracked while the camera is moved, and reconstructed in 3-D using the procedure described in section 4. Visual information (intersection line between the planes, coordinate system axes) are displayed on the video stream in order to help the user to evaluate the accuracy of the solution and decide to cancel or validate it.

Camera tracking. Once some 3-D blobs are available in the map, multi-planar camera tracking can perform according to section 5 and virtual objects can be added to the scene.

Map expansion. At any time of the process, new 3-D blobs can be added to the map using the camera-mouse. These blobs can give rise to new planes or expand existing ones (section 6).

Failure recovery. A procedure is used to recover from tracking failure, e.g. due to fast camera motion (section 7). This procedure is based on SIFT feature matching[3] between a set of keyframes and the current frame. The keyframes are stored during the working session upon user request and each time a 3-D blob is added to the map.

Bundle adjustment. A bundle adjustment of the position and orientation of all the planes and keyviews in the map can be requested at any time of the process, as described in section 8.

Intersection lines with the reference plane play a crucial role in this process. Given the correspondence of image lines between a pair of images, the homographies induced by planes containing the line in 3-space are reduced from a 3-parameter to a one-parameter family [24]. This property is used there to get faster convergence and improved accuracy during the map building steps. Moreover, intersection lines provide visual hints that help the user to assess the accuracy of the system-generated results and prevent map corruption.

4 Map Initialization

When two non-parallel planes are observed from two different views, closed-form solutions exist for computing the equation of the planes as well as the camera motion between the views [6,33]. However, the accuracy of these methods is generally too poor for applications where the re-projected structures have to be perfectly aligned with their image counterpart. This is partly due to the fact that, although the same camera motion is applied to the two planes, motion parameters are computed separately for each plane. Two solutions are obtained, which are generally similar but not identical, and one of these solutions has to be arbitrary chosen. It is also not possible to integrate constraints such as fixing the angle between the planes. For similar reasons, closed-form solutions are not well adapted to handle more than two views of the same planes. Closed-form solutions are therefore mainly used as initial guesses to minimize the difference between some observed image points and their re-

[1] Harris corners [10] are accurate, fast to compute and easy to match between subsequent frames using cross-correlation.

[2] `http://www.loria.fr/~gsimon/vc`

[3] SIFT features [21] are invariant to image scale and rotation and are shown robust to some extend to affine distortion, change in viewpoint and change to illumination. This makes them particularly suitable for feature matching between distant images.

projections so that the estimation is optimized in the least-square sense [33].

In [13], an extended Kalman filter (EKF) is used to causally estimate the camera motion, as well as the equation of the planes related to some planar patches and the parameters of an affine model used to handle the illumination changes on the patches. However, Kalman filtering is sensitive both to the initial values of the state vector and the tuning of the first covariance matrices. Moreover, the size of the state vector is relatively high (18-parameters for two planes) due to the fact that the planar patches are tracked inside the EKF framework.

We propose to divide this computationally complex estimation problem into several simpler ones:

1. the planar surfaces are tracked independently between the images, using RANSAC matching of Harris corners;
2. the projected intersection line between the observed planes (two parameters) is causally estimated using a particle filter (PF). PF has these advantages over EKF, that it does not require any initial estimate of the state vector and it can handle non-linear measurement functions. Moreover, the PF framework allows to easily fuse different kinds of likelihood measurements: in our case, a geometric constraint imposed by the homographies and a photometric constraint due to the specific appearance of an intersection line in the image can be considered together, enhancing both the rate and the accuracy of the detection;
3. the motion of the camera and the equation of the planes in the 3-D Euclidean space are linearly approximated and then iteratively refined using the Levenberg-Marquardt optimization method. As the projection of the intersection line in the first image is known, the equations of the two planes can be expressed using 3 instead of 5 parameters. In addition to reducing the risk of being trapped into a local minimum and shortening the processing time of the iterative optimization, this simpler parameterization allows to easily incorporate knowledge on the angle between the planes.

Each stage of this algorithm is thus a rather simple task, and an important point is that the intermediate results can be assessed by the system or the operator before going to the next stage:

- tracking failures in stage 1 are automatically detected by simply thresholding the number of inlier matches of the homographies (see section 5),
- the detected line of stage 2 can be visually assessed by simple image comparison,
- the 3-D reconstruction obtained at stage 3 can be visually assessed by displaying the world coordinate frame; in order to facilitate this assessment, the origin of the coordinate frame is put on the middle of the intersection line and one of the three axes is aligned with that line.

Sections 4.1 and 4.2 now detail how the causal estimation of the projected intersection line is performed, which is a crucial stage of the algorithm; the Euclidean reconstruction of the scene - camera geometry based on the knowledge of that line is then explained in section 4.3.

4.1 Preliminaries

We first set out some theoretical results that will be useful. A plane projective transformation is a planar *homology* if it has a line of fixed points (the *axis*), together with a fixed point not on the line (the *vertex*) [14, 11]. Algebraically, an equivalent statement is that the 3×3 matrix representing the transformation has two equal and one distinct eigenvalues. The axis is the join of the eigenvectors corresponding to the degenerate eigen-values. The third eigenvector corresponds to the vertex.

Suppose we have two images, I_1 and I_2, of a scene consisting of two non-parallel planes, π_1 and π_2. Let $\mathbf{C}_1$ and $\mathbf{C}_2$ be the positions of the camera center when I_1 and (resp.) I_2 were acquired. We further assume that the full-rank planar homographies, $\mathbf{H}_1$ and $\mathbf{H}_2$, induced by planes π_1 and (resp.) π_2 between I_1 and I_2 are known.

Proposition 1. *The* 3×3 *matrix* $\mathbf{S} = \mathbf{H}_2^{-1}\mathbf{H}_1$ *is a planar homology, whose axis is the projection* $\mathbf{l}$ *in* I_1 *of the intersection line between* π_1 *and* π_2 *and the vertex is the epipole* $\mathbf{e}$*, projection of* $\mathbf{C}_2$ *in* I_1.

Proof Let us first prove that any point on line $\mathbf{l}$ is fixed by $\mathbf{S}$. Let $\mathbf{p}$ be a point on $\mathbf{l}$. $\mathbf{p}$ is the projection in I_1 of a point $\mathbf{P}$ on the intersection line between π_1 and π_2. As $\mathbf{P}$ is on π_1, $\mathbf{p}$ is transformed by $\mathbf{H}_1$ to the projection $\mathbf{p}'$ of $\mathbf{P}$ in I_2. Now as $\mathbf{P}$ is on π_2, $\mathbf{p}'$ is transformed by $\mathbf{H}_2^{-1}$ to the projection $\mathbf{p}$ of $\mathbf{P}$ in I_1. This yields $\mathbf{H}_2^{-1}\mathbf{H}_1\mathbf{p} \sim \mathbf{p}$ where $\sim$ denotes an equality up to a scale factor.

Let us now prove that the epipole $\mathbf{e}$ is fixed by $\mathbf{S}$. $\mathbf{e}$ is the projection in I_1 of a point $\mathbf{P}_1$ on π_1. $\mathbf{P}_1$ is the intersection between π_1 and the ray passing though $\mathbf{C}_1$ and $\mathbf{C}_2$. As $\mathbf{C}_1$, $\mathbf{C}_2$ and $\mathbf{P}_1$ are aligned, the projection $\mathbf{e}' = \mathbf{H}_1\mathbf{e}$ of $\mathbf{P}_1$ in I_2 is the epipole in I_2. Using the same reasoning but reversing the roles of I_1 and I_2, we prove that $\mathbf{H}_2^{-1}\mathbf{e}' = \mathbf{H}_2^{-1}\mathbf{H}_1\mathbf{e}$ is the epipole $\mathbf{e}$ in $\mathbf{I}_1$, which concludes the proof.

Corollary 1. *The* 3×3 *matrix* $\mathbf{T} = \mathbf{H}_2^T\mathbf{H}_1^{-T}$ *is a planar homology, whose axis is a pencil of lines intersecting at the epipole* $\mathbf{e}$ *and the vertex is the projection* $\mathbf{l}$ *of the intersection line between* $\pi_\mathbf{1}$ *and* $\pi_\mathbf{2}$.

Proof As $\mathbf{T} = \mathbf{S}^{-T}$, this result can be obtained by applying the principle of duality between points and lines in the projective plane $\mathbb{P}^2$ to Proposition 1: the "set of collinear points" forming the axis of $\mathbf{S}$ is the same as the "set of concurrent lines" forming the axis of $\mathbf{T}$ in the dual space. Similarly, the "distinct fixed point" forming the vertex of $\mathbf{S}$ becomes the "distinct fixed line" forming the vertex of $\mathbf{T}$. Finally, just as the collinear fixed points of $\mathbf{S}$ form the distinct fixed line $\mathbf{l}$ of $\mathbf{T} = \mathbf{S}^{-T}$, the concurrent fixed lines of $\mathbf{T}$ intersect at the distinct fixed point $\mathbf{e}$ of $\mathbf{S} = \mathbf{T}^{-T}$.

Corollary 2. *When applying the projective transformation* $\mathbf{S}$*, a point on line* $\mathbf{l}$ *is mapped to itself, whereas a point on a line passing through* $\mathbf{e}$ *is generally mapped to another point on the line: among the fixed lines of* $\mathbf{T}$*, only* $\mathbf{l}$ *is fixed pointwise.*

Proof This directly comes from Proposition 1: the only fixed points of $\mathbf{S}$ are the points on $\mathbf{l}$ and the epipole $\mathbf{e}$. A geometrical interpretation of this result is shown in figure 1: $\mathbf{p}' = \mathbf{S}\mathbf{p}$ is generally distinct from $\mathbf{p}$, except when $\mathbf{p} = \mathbf{e}$ or $\mathbf{p}$ is on line $\mathbf{l}$.

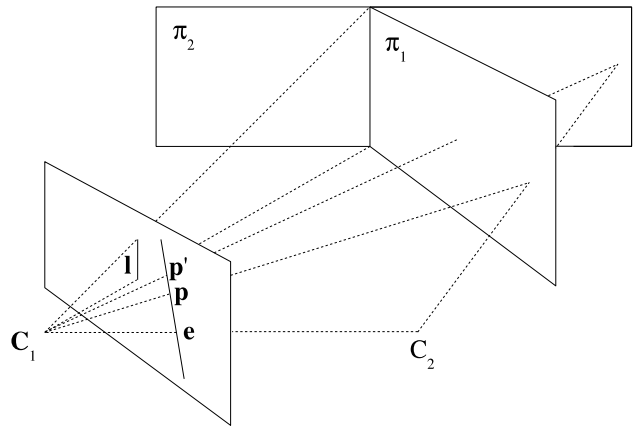


Fig. 1 Fixed lines of the homology.

4.2 Detection of the intersection line

We describe in this section a major contribution of our approach, that consists in using temporal geometric and (optionally) photometric consistency to detect the intersection lines between the reference plane and other non-parallel planes.

We now assume we are able to compute several pairs of homographies $\mathbf{H}_1^i$, $\mathbf{H}_2^i$ between I_1 and subsequent images I_i. From Corollary 1, the imaged intersection line $\mathbf{l}$ may be algebraically computed as the distinct eigenvector of any homology $\mathbf{T}^i = \mathbf{H}_2^i {\mathbf{H}_1^i}^{-T}$. However, this is unstable in practice as $\mathbf{T}^i$ are non-symmetric matrices. By contrast, $\mathbf{l}$ can be reliably computed using temporal consistency. We use particle filtering, a well known technique for implementing a recursive Bayesian filter by Monte Carlo simulations [1]. The key idea is to represent the required posterior density function $p(\mathbf{x}_i|\mathbf{z}_{1:i})$, where $\mathbf{z}_{1:i}$ is the set of all available measurements up to time i, by a set of random samples $\mathbf{x}_i^j$ with associated weights w_i^j, and to compute estimates based on these samples and weights:

$$p(\mathbf{x}_i|\mathbf{z}_{1:i}) \approx \sum_{j=1}^{N} w_i^j \delta(\mathbf{x}_i - \mathbf{x}_i^j), \sum_{j=1}^{N} w_i^j = 1. \quad (1)$$

We implement the generic PF according to the framework described in [1]. Resampling is used whenever a significant degeneracy is observed (i.e., when the effective sample size N_{eff} falls below some threshold N_T). Our implementation has the following characteristics:

1. particles are homogeneous coordinates of lines under the form $[\cos(\theta), \sin(\theta), -\rho]^T$. The first mode of distribution (1) is taken as the estimated line $\mathbf{l}$ (in image I_1) at time i. Initially, the particles are uniformly distributed inside the largest ellipse $\mathcal{E}$ contained in the image (see Fig.2, first frame);
2. the prior $p(\mathbf{x}_i|\mathbf{x}_{i-1})$ is the normal distribution centered at $\mathbf{x}_{i-1}$ with covariance matrix $\mathbf{V}$ ($\mathbf{V} = \mathbf{I}[10^{-4}, 10^{-4}, 25]^T$ in our experiments); the importance density is the prior;
3. the likelihood density at time i is given by:

$$p(z_i|\mathbf{x}_i) = p(z_i^g|\mathbf{x}_i)p(z_i^p|\mathbf{x}_i),$$

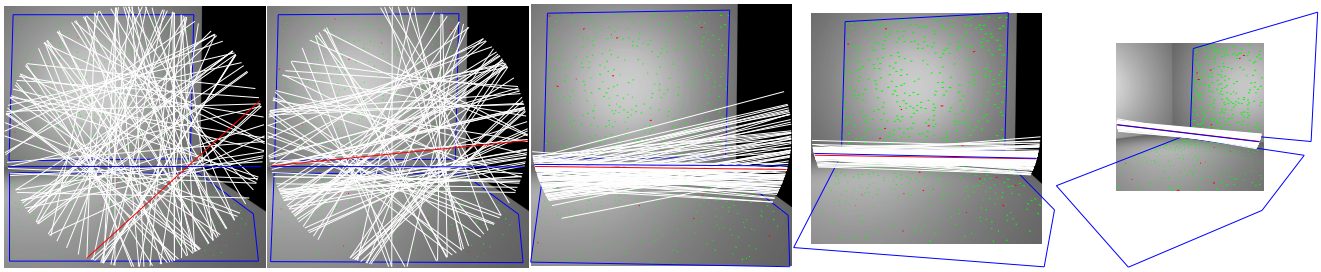
where z_i^g and z_i^p are (assumed independent) geometric and (resp.) photometric measurements we now detail.

4.2.1 Geometric likelihood

Measuring "how fixed" a line is when transformed by $\mathbf{T}^i$ provides a geometric measure of the likelihood of the related particle. However, it has been shown in section 4.1, Corollary 1, that an infinity of lines are fixed by $\mathbf{T}^i$: the vertex $\mathbf{l}$ of $\mathbf{T}^i$, but also any line passing through the epipole of the camera associated with image I_i in image I_1. In order to avoid confusion between all these possible candidates, we use Corollary 2 and measure the fixity of some points on the line rather than the global fixity of the line. In practice, we found enough discriminant to measure the fixity of the intersection points $\mathbf{p}_1$ and $\mathbf{p}_2$ of the line with the ellipse $\mathcal{E}$:

$$p(z_i^g|\mathbf{x}_i) \propto \exp\left(-\frac{D^2}{2\sigma_g^2}\right), D = \sqrt{\frac{1}{2}\sum_{k=1}^{2} \|z(\mathbf{p}_k) - z(\mathbf{S}^i\mathbf{p}_k)\|^2}$$

where $\|.\|$ denotes the L2 norm of a vector, $z : \mathbb{P}^2 \to \mathbb{R}^2$ is a function such that $z((x\ y\ z)^T) = (x/z\ y/z)^T$, $\mathbf{S}^i = {\mathbf{H}_2^i}^{-1}\mathbf{H}_1^i$ and σ_g is set to 3 in our experiments.



frame #1 frame #5 frame #10 frame #20 frame #50

Fig. 2 Particle filtering on a synthetic sequence.

4.2.2 Photometric likelihood

Accuracy and convergence of the PF can be increased by measuring the distance of the particles to the highest gradients of the image. This is done by applying the following operations to the images:

1. Sobel filtering,
2. hysteresis thresholding,
3. lines detection using the Hough Transform (HT).

The HT is accelerated by only incrementing accumulators (ρ, θ) for which θ is compatible in a range of ± 5 degrees with the gradient vector obtained by Sobel filtering. A significant pruning is also obtained by computing a single global HT updated from frame to frame by transferring the line candidates of image i to image 1, using the homography $\mathbf{H}_k^{i\,-T}$, $k = 1$ or $k = 2$: doing that, only the projections of the lines on plane π_k contribute to the local maxima of the HT (other lines are transferred to varying coordinates of the HT). Finally, we keep the lines $\mathbf{m}_i^j$ corresponding to the M greatest local maxima of the HT ($M = 100$ in our experiments). This leads to:

$$p(z_i^p|\mathbf{x}_i) \propto \exp\left(-\frac{D^2}{2\sigma_p^2}\right), D = \min_{j=1}^{M} \sqrt{\frac{1}{2}\sum_{k=1}^{2}(\mathbf{m}_i^j|\mathbf{p}_k)^2}, \qquad (2)$$

where $(.|.)$ denotes the dot product, $\mathbf{m}_i^j$ is expressed under the form $[\cos(\theta), \sin(\theta), -\rho]^T$ and $\sigma_p = \sigma_g$ in our experiments. This measure benefits from the robustness of the HT and can therefore tolerate partial occlusions of the intersection line (see for instance the results presented in section 9.2).

In our system, the intersection line starts to be estimated and displayed as soon as a new 2-D blob is introduced on a non-reference plane while another 2-D blob is already being tracked on the reference plane, or vice versa. The two blobs can be extended during the particle filtering process. The user can choose between using both the geometric and the photometric measurements or the geometric measurement alone in case where the intersection line is occluded by some physical objects (see section 9.2). In case of slow convergence, the user can reset the particle filter using the keyboard left key. Once the user validates the estimated line (using the right key), the Euclidean reconstruction procedure we now describe is called.

4.3 Euclidean reconstruction

The reconstruction procedure in image I_i makes use of:

- the projection $\mathbf{l}$ in image I_1 of the intersection line between two planes π_1 and π_2
- a pair of homographies $\mathbf{H}_1^i$ and $\mathbf{H}_2^i$ (below denoted as $\mathbf{H}_1$ and $\mathbf{H}_2$ for sake of simplicity) that map points lying on π_1 (resp. π_2) between I_1 and I_i.

Moreover, we assume that the intrinsic parameters of the camera are known and that the homogeneous coordinates of the image points are affinely pre-transformed using the inverse intrinsic matrix [6]. From these data, we want to get the equations $\mathbf{\Pi}_1 = [\mathbf{n}_1^T, d_1]^T$ and $\mathbf{\Pi}_2 = [\mathbf{n}_2^T, d_2]^T$ of π_1 and (resp.) π_2 in the first view coordinate system, as well as the camera motion $\mathbf{R}, \mathbf{t}$ between I_1 and I_i. As d_1 is an overall scale factor[4], and as we can use $\|\mathbf{n}_1\| = \|\mathbf{n}_2\| = 1$, we have to estimate 11 parameters: 2 for π_1, 3 for π_2, 3 for $\mathbf{R}$ and 3 for $\mathbf{t}$.

However, as $\mathbf{l}$ is known, it is shown in Fig. 1 that π_2 belongs to a sheaf of planes passing through the 3-D intersection line between π_1 and the plane passing through $\mathbf{l}$ and the camera center $\mathbf{C}_1$. This is algebraically expressed as [11]:

$$\mathbf{\Pi}_2 \sim \mathbf{\Pi}_1 + \lambda[\mathbf{l}^T 0]^T, \qquad (3)$$

where λ is an unknown scalar. Therefore, we can replace the three parameters used to describe $\mathbf{n}_2, d_2$ by only one additional parameter λ, and finally estimate 9 parameters instead of 11. The benefits of this parameter reduction in terms of accuracy and computation times are illustrated in section 9.1. Moreover, in the common case where π_1 is orthogonal to π_2, equation $(\mathbf{n}_1|\mathbf{n}_2) = 0$ coupled with equation (3) lead to $\lambda = -1/(\mathbf{n}_1|\mathbf{l})$ and only 8 parameters to be estimated.

[4] in our implementation, d_1 is set to the assumed distance between the camera center at the beginning of the process and the reference plane.

Each set of values for the 9 or 8 parameters gives rise to two predicted homographies $\{\hat{\mathbf{H}}_j\}_{j\in\{1,2\}}$ given by [6]:

$$\hat{\mathbf{H}}_j = d_j\mathbf{R} + \mathbf{t}\mathbf{n}_j^T.$$

Therefore, if $n_1 \geq 4$ points $\{\mathbf{v}_1^k\}_{1\leq k\leq n_1}$ and $n_2 \geq 4$ points $\{\mathbf{v}_2^k\}_{1\leq k\leq n_2}$ are lying on π_1 and (resp.) π_2 in I_1, the 9 or 8-parameter set $\mathbf{p}$ can be estimated by iterative minimization of the mean square distance between the points transformed by $\hat{\mathbf{H}}_j$ and the points transformed by $\mathbf{H}_j$:

$$\mathbf{p} = \text{argmin} \sum_{i=1}^{2}\sum_{k=1}^{n_i} \|z\left(\hat{\mathbf{H}}\mathbf{v}_i^k\right) - z\left(\mathbf{H}\mathbf{v}_i^k\right)\|. \quad (4)$$

In our system, the points $\{\mathbf{v}_i^k\}$ are the vertices of the 2-D blobs defined by the user. Minimization (4) is performed using the Levenberg-Marquardt algorithm. This algorithm requires initial values for $\mathbf{p}$. It is shown in [6] that the simultaneous estimate of the camera motion and the plane pose corresponding to a homography has in general two physical solutions which can be obtained using SVD. As we know two homographies, this twofold ambiguity can be removed by finding the common solution for camera motion: this provides initial values for $\mathbf{R}, \mathbf{t}$ and the equations of the planes.

As for the intersection line detection step, the user has the opportunity to validate the solution, in which case two 3-D blobs are added to the map, or to reset the procedure, in which case I_1 is set to the current frame.

5 Camera Tracking

Once the initialization stage has been done, the added 3-D blobs can be used to compute the camera pose from frame to frame. Formally, a 3-D blob b is a 3-D planar polygon included in a plane π_b of equation $[X, Y, Z, 1]\Pi_b = 0$, where $\Pi_b = [\mathbf{n}_b^T, d_b]^T$ is expressed in the world coordinate system. It must be noted that several blobs can share the same plane equation. A 3-D blob is said *visible* relatively to a camera pose $\mathbf{R}, \mathbf{t}$ (given some camera intrinsic parameters and an image size) if it is in front of the camera and a part of its surface projects inside the image and is not totally occluded by another 3-D blob. This visibility test can be done efficiently using the depth test and stencil buffer available in most graphics rendering engines. In the following, $\mathcal{V}(\mathbf{R}, \mathbf{t})$ denotes the set of 3-D map blobs that are visible relatively to the camera pose $\mathbf{R}, \mathbf{t}$.

Camera poses are computed from inter-image tracking of the blobs, according to the method described in [31]. At every frame, the system performs the following two-stage procedure:

1. Let $\tilde{\mathbf{R}}, \tilde{\mathbf{t}}$ be the camera pose obtained in the previous frame. For each 3-D blob in $\mathcal{V}(\tilde{\mathbf{R}}, \tilde{\mathbf{t}})$, a set of point matches is determined using normalized cross-correlation between Harris corners in the current frame and Harris corners inside the visible part of the blob in the previous frame; this set of matches is obtained using a RANSAC-based computation of the planar homography that best fits the corners between the two views; blobs that do not get enough matches are not considered further in this iteration; in the case where no blob remains, the failure recovery procedure described in section 7 is called.
2. The camera pose $\mathbf{R}, \mathbf{t}$ is computed by iterative minimization of the global transfer error of the visible blobs:

$$\mathbf{R}, \mathbf{t} = \text{argmin} \sum_{b\in\mathcal{V}(\tilde{\mathbf{R}},\tilde{\mathbf{t}})} MSE(\Pi_b, \tilde{\mathbf{R}}, \tilde{\mathbf{t}}, \mathbf{R}, \mathbf{t}), \quad (5)$$

where $MSE(\Pi_b, \tilde{\mathbf{R}}, \tilde{\mathbf{t}}, \mathbf{R}, \mathbf{t})$ is the mean square residual error computed over the corner correspondences obtained for blob b. The residual error of a pair of corners is the Euclidean distance between the corner in the current frame and the corner in the previous frame transferred to the current frame using the induced homography [6]:

$$\mathbf{H}(\Pi_b, \tilde{\mathbf{R}}, \tilde{\mathbf{t}}, \mathbf{R}, \mathbf{t}) = \mathbf{R}\tilde{\mathbf{R}}^T - \frac{(\mathbf{t} - \mathbf{R}\tilde{\mathbf{R}}^T\tilde{\mathbf{t}})(\tilde{\mathbf{R}}\mathbf{n}_b)^T}{(d_b - (\tilde{\mathbf{t}}|\tilde{\mathbf{R}}\mathbf{n}_b))}. \quad (6)$$

This 6-parameters non-linear optimization is performed using the Levenberg-Marquardt algorithm, initial values being provided by $\tilde{\mathbf{R}}, \tilde{\mathbf{t}}$.

At the end of step 2, Akaike's model selection criterion is used to prevent unpleasant jittering effects induced by image noise [31]. This criterion is based on a balancing of the residual error obtained from minimization (5) and a complexity term which penalizes higher order motion models. If a stationary motion is detected, the current pose is replaced by the previous one; in the case where a pure rotation is detected, minimization (5) is performed one more time under the constraint $\mathbf{t} = \tilde{\mathbf{t}}$.

6 Map Expansion

During the tracking process, new 2-D blobs can be defined using the camera-mouse. In the case where a blob is drawn on the reference plane, say π_1, its 3-D counterpart is obtained by back-projecting the related 2-D vertices onto the known reference plane. In the case where a blob is drawn on a plane non-parallel to the reference plane, the intersection line with the reference plane is filtered. If this line is aligned with another intersection line in the map, merging with the related plane is proposed by drawing the involved blobs in a special color (green on the video). This proposal can be accepted by pressing the right key. Pressing the right key when no merging was proposed involves adding a new plane to the map. In that case, the equation $\mathbf{\Pi}_2$ of the new plane is obtained using a similar procedure as in section

4.3, except that, as the equation $\mathbf{\Pi}_1$ of the reference plane is known as well as the camera pose $\mathbf{R}, \mathbf{t}$, we only have to iteratively compute one parameter (λ). If the new plane is known to be perpendicular to the reference plane, we even get a closed-form solution for λ as explained in section 4.3.

7 Failure Recovery

A keyframe k is stored in memory upon user request and each time a new 3-D blob is added to the map. A keyframe is a data structure containing the camera pose $\mathbf{R}_k, \mathbf{t}_k$ computed when the keyframe was stored and a set of SIFT features extracted from the related frame and stored in a k-d tree. Only features inside a visible blob in the keyframe are stored, accompanied with a link to the related 3-D blob. When a keyframe is introduced due to the addition of a 3-D blob b to the map, a link $k(b)$ to that keyframe is also stored in the data structure associated to the 3-D blob, as well as the set of Harris corners and associated patches detected inside the corresponding 2-D blob (see Fig. 3). These corners will be used to refine the SIFT-based pose estimate as described below. When a tracking failure is detected, the following two-stage process is performed (Fig. 3):

1. *SIFT-based rough pose estimate.* SIFT features are detected in the current frame and RANSAC matching is performed with the SIFT features extracted inside each blob visible in keyframe k ($k = 1$ initially); if a sufficient number of matches is found inside at least one of these blobs, the current camera pose is computed by performing minimization (5) between the keyframe and the current frame, using all the blobs visible in keyframe k containing enough matches; otherwise, k is incremented and this stage is repeated until a keyframe is found or the list of keyframes is empty (which means that the recovery procedure failed for the current frame).
2. *Pose refinement.* The first stage provides an approximate camera pose $\hat{\mathbf{R}}, \hat{\mathbf{t}}$ for the current frame from which camera tracking may resume. However, this pose was obtained using the blobs visible both in the current frame and the selected keyframe, whereas other blobs can be visible in the current frame that may be used to refine the pose. Visible blobs can be identified using the approximate camera pose $\hat{\mathbf{R}}, \hat{\mathbf{t}}$. For each blob visible in the current frame, an estimate of the homography that maps this blob between the keyframe where it was introduced and the current frame can be obtained: $\hat{\mathbf{H}} = \mathbf{H}(\mathit{\Pi}_b, \mathbf{R}_{k(b)}, \mathbf{t}_{k(b)}, \hat{\mathbf{R}}, \hat{\mathbf{t}})$. This homography can be used to generate a set of reliable matches $(x^i_b, y^i_b) \leftrightarrow (x'^i_b, y'^i_b)$, referred to as *H-matches* in the following, between the keyframe and the current frame: (x^i_b, y^i_b) is the Harris corner i stored in the data structure associated to the blob; this corner is transferred to the current frame using $\hat{\mathbf{H}}$, and a cross-correlation score is computed between all the patches in a neighborhood of the transferred corner (typically, a search window of size 11) and the patch associated to the corner in the keyframe, warped using $\hat{\mathbf{H}}$. (x'^i_b, y'^i_b) is taken as the center of the patch that obtains the highest score. Outliers can be removed using the RANSAC paradigm. Finally, the following non-linear optimization is performed using the generated sets of H-matches:

$$\mathbf{R}, \mathbf{t} = \text{argmin} \sum_{b \in \mathcal{V}(\hat{\mathbf{R}}, \hat{\mathbf{t}})} MSE(\mathit{\Pi}_b, \mathbf{R}_{k(b)}, \mathbf{t}_{k(b)}, \mathbf{R}, \mathbf{t}), \qquad (7)$$

with initial values of $\mathbf{R}, \mathbf{t}$ set to $\hat{\mathbf{R}}, \hat{\mathbf{t}}$.

8 Bundle Adjustment

When a new keyframe k is introduced, a set of H-matches is computed for each visible blob $b \in \mathcal{V}(\mathbf{R}_k, \mathbf{t}_k)$ introduced previously ($k(b) < k$), between the current keyframe k and the original keyframe $k(b)$. Therefore, at any time of the process, poses of the N_K keyframes as well as the plane equations of the N_B 3-D blobs can be refined by performing a global bundle adjustment:

$$\{\mathbf{R}_k, \mathbf{t}_k\}_{1 \leq k \leq N_K}, \{\mathit{\Pi}_b\}_{1 \leq b \leq N_B} = \text{argmin} \sum_{k=2}^{N_K} \sum_{b \in \mathcal{V}(\mathbf{R}_k, \mathbf{t}_k), k(b) < k} MSE(\mathit{\Pi}_b, \mathbf{R}_{k(b)}, \mathbf{t}_{k(b)}, \mathbf{R}_k, \mathbf{t}_k). \qquad (8)$$

In our system, this optimization is done on the fly upon user request: as the blobs are generally wide areas of the scene, their number is relatively small (typically, 5 to 20 blobs are enough to define a room), which makes the process much faster than traditional bundle adjustments (examples of computation times are given in table 1). However, as camera tracking can perform from the initial map while the bundle adjustment is running, we could as in [17] split tracking and bundle adjustment into two separately-scheduled tasks, running on different processing cores.

9 Results

9.1 Synthetic results

Synthetic results were obtained using a 80-frame sequence, in which the camera follows a circular path while pointing toward a horizontal and a vertical plane (Fig. 2). A Gaussian noise of standard deviation 0.3 was added to the coordinates of the points used to compute the inter-image homographies. Fig. 2 shows examples of particle distributions obtained in several images of the sequence.

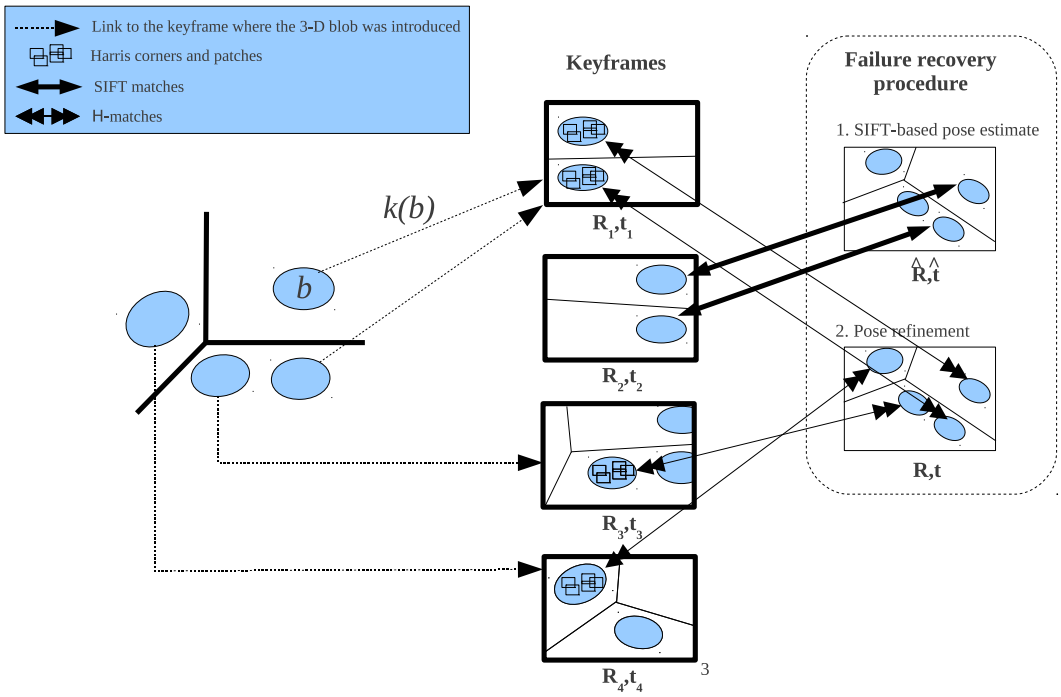


Fig. 3 Failure recovery process.

9.1.1 Filtering parameters

Some tests have been performed in order to assess the effects of the number of particles N and the resampling threshold N_T over the convergence of the PF. Figure 4 shows the mean number of frames (over 100 tests) needed to reach the convergence, for N varying from 20 to 1000 and N_T from 0 to N. The convergence is always reached, even for small values of N (except for $N_T = 0$ due to the degeneracy problem). However, the convergence is faster for high values of N and when N_T is closer to N. These results led us to use $N = 1000$ particles in our real experiments, which allows fast convergence without compromising global performance and $N_T = N$, which means resampling is performed at each frame.

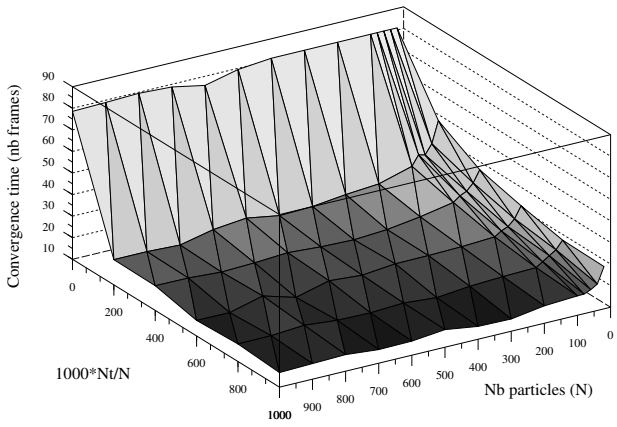


Fig. 4 Convergence time of the particle filter for different values of N (number of particles) and N_T (resampling threshold).

9.1.2 Euclidean reconstruction

Figure 5 shows the errors obtained on the normal to the horizontal plane and the x-coordinate of the camera translation when computing structure and motion between the first frame of the synthetic sequence and the next 50 frames. Errors are shown for the SVD, the 11-parameters optimization and the 9-parameters optimization (in that case the intersection line is extracted from a Hough transform obtained in the first frame). This graphic shows that the 9-parameters optimization can substantially improve the accuracy, especially when the baseline is small (except for too small baselines which lead to unstable results). Moreover, the mean number of iterations of the Levenberg-Marquardt algorithm is 3.9 for the 9-parameters optimization, against 6.7 for the 11-parameters optimization.

9.2 An miniature scene

Convergence of the particle filter in presence of cluttered objects is illustrated on a miniature interior scene (Fig. 6). Two experiments were performed: in the first experiment,

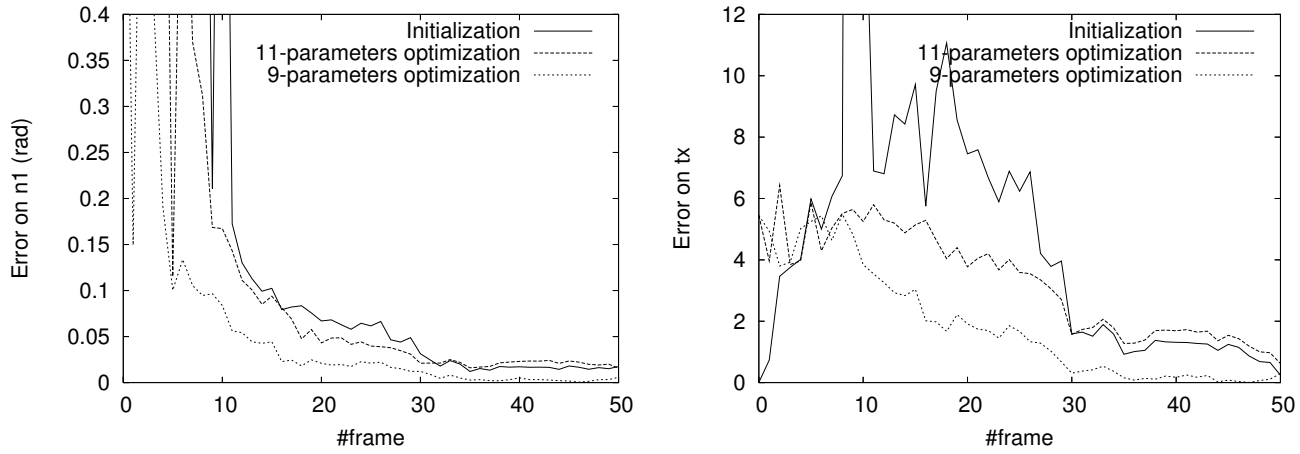


Fig. 5 Structure (left) and motion (right) errors measured on the synthetic sequence.

the intersection line between the ground plane and a wall is occluded by several objects but still partly visible. The geometric as well as photometric measurements are used in particle filtering. A video on our website shows the convergence of the particle filter at initialization stage and during a plane expansion operation[5]. As shown at top of figure 6, the convergence at initialization is very accurate and the second blob is well recognized as being on the same plane as the first blob. This illustrates the robustness of the Hough transform that is used in photometric measurements.

In the second experiment, the intersection line is almost completely occluded and only the geometric constraint is used in particle filtering. The same operations are done as in the first experiment (see the second video on the website). As shown at bottom of figure 6, the detected line is less accurate than previously but the error is small and again, the second blob can be merged with the first one.

9.3 An indoor scene

Real-size experiments were realized inside a two-room scene (Fig. 7). The system ran at about 12 Hz in tracking mode and 8 Hz in tracking + filtering mode on a PC Dell Precision 390, 2.93 Ghz, while part of the processor was used to capture a video of the screen. A Sony DFW-VL500 camera was used at resolution 320x240; the PC was placed on a trolley, wheeled around the scene, while the camera was held by hand. The session, which lasted about 10 minutes, was captured in a 8000-frame sequence; a video is associated to the paper that shows the most interesting parts of that sequence. The right window of the video shows a 3-D view of the map and camera motions.

[5] In this video, the line particles are displayed to demonstrate the convergence of the particle filter. These lines are not shown during standard use of the application.

During the session, 13 blobs were defined on 6 planes (including the floor), and 18 keyframes were introduced. Each time a horizontal blob is added to the map, a virtual robot is added on the blob; similarly, virtual cubes containing the number of the associated plane are added on each vertical blob (the orthogonality constraint between the floor and the walls is used by the system). As one can see in the video, the virtual objects appear firmly anchored in the scene, and camera tracking performs well despite erratic motions of the hand-held camera. The failure recovery procedure is called 5 times, either due to an abrupt motion of the camera, or because the known part of the scene disappeared from the field of view.

Table 1 shows the error angles between the first vertical plane added to the map and the other vertical planes, obtained after each major mapping operation; these errors were computed considering the planes are orthogonal each other. The initial estimate of the second plane has a poor accuracy (10.3 deg error); a bundle adjustment based on $N_k = 6$ keyframes and $N_B = 3$ blobs is first attempted, which does not significantly improve the accuracy; one more keyframe and a new bundle adjustment allow to obtain satisfying accuracy (1.5 deg). The next three planes are successively added without performing new bundle adjustments; plane 4 is initialized with 12.6 deg error, which is due to the fact that this plane appears in fronto-parallel position and contains few texture information (see Fig. 7, frame #5749). However, a final bundle adjustment reduces this error to 0.6 deg (see the last row of table 1 for the other final errors); the accuracy of plane 5 is not improved by this last operation due to the fact that only one keyframe was added since that plane was introduced.

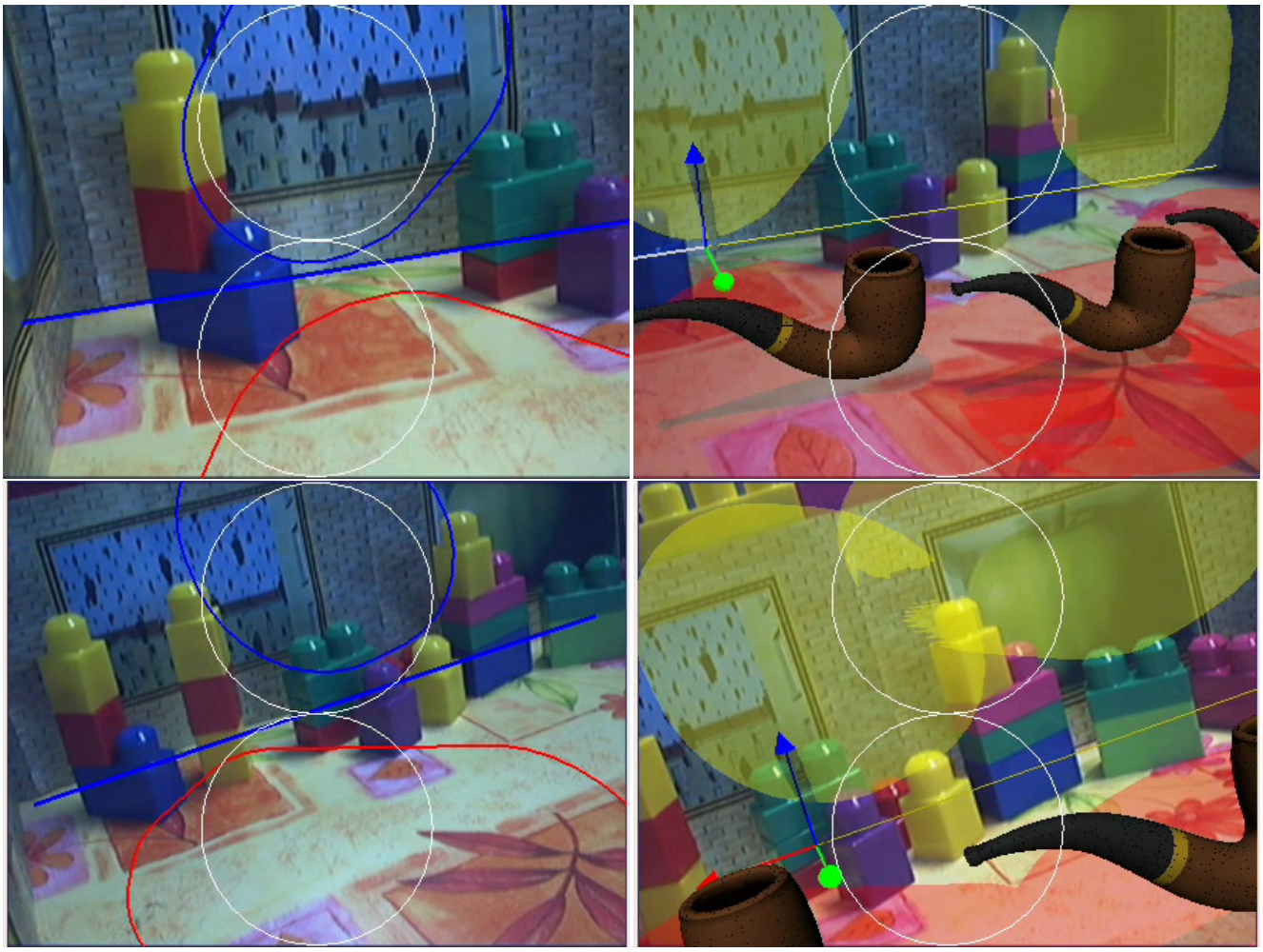


Fig. 6 Convergence of the particle filter. Top: geometric and photometric measurements are used, as the intersection line is still partly visisble. Bottom: more occluding objects are added and only the geometric measurement is used.

#frame	N_K	N_B	Event	(1,2)	(1,3)	(1,4)	(1,5)
675	3	3	P2 added	-10.3			
792	6	3	Bundle (0.3 s)	9.1			
818	6	3	Undo bundle	-10.3			
891	7	3	Bundle (0.3 s)	1.5			
2710	14	10	P3 added	1.5	-1.5		
2954	16	11	P4 added	1.5	-1.5	12.6	
5648	17	12	P5 added	1.5	-1.5	12.6	7.2
7842	18	13	Bundle (2.1 s)	0.9	-3.2	0.6	7.3

Table 1 Mapping operations and angle errors (in deg) between the vertical planes along the real-size sequence.

9.4 An outdoor scene

In order to assess the performance of the algorithm in a more natural scene, we have tested our system in an outdoor environment. Figure 8 shows some snapshots of the session and the complete video sequence is available on our website. The reference plane is taken as the ground plane which is covered by grass. Two perpendicular but poorly textured walls are also visible. Despite this, we succeeded in defining blobs on these walls by exploiting the ability for the user to focus on relevant parts of the scene - here pipes and a door - to build and extend the blobs. Due to the joint use of texture - in homography computation - and intersection line information, detection of the vertical walls is really good. The error angle between these planes is 3.6 deg.

To estimate whether it would be possible to reconstruct and segment automatically this scene, we have used the Voodoo Camera Tracker [27]. This non-commercial software estimates camera parameters and reconstructs a 3-D scene from image sequences. Several algorithms are available for feature detection and tracking. The structure of the scene as well as the camera motions are computed sequentially, and a final bundle adjustment is performed once all the frames have been solved. In order to be as fair as possible, we did not use the sequence recorded during the working session. Actually, all the algorithms failed solving this sequence, which is not really surprising as it contains camera motions, includ-

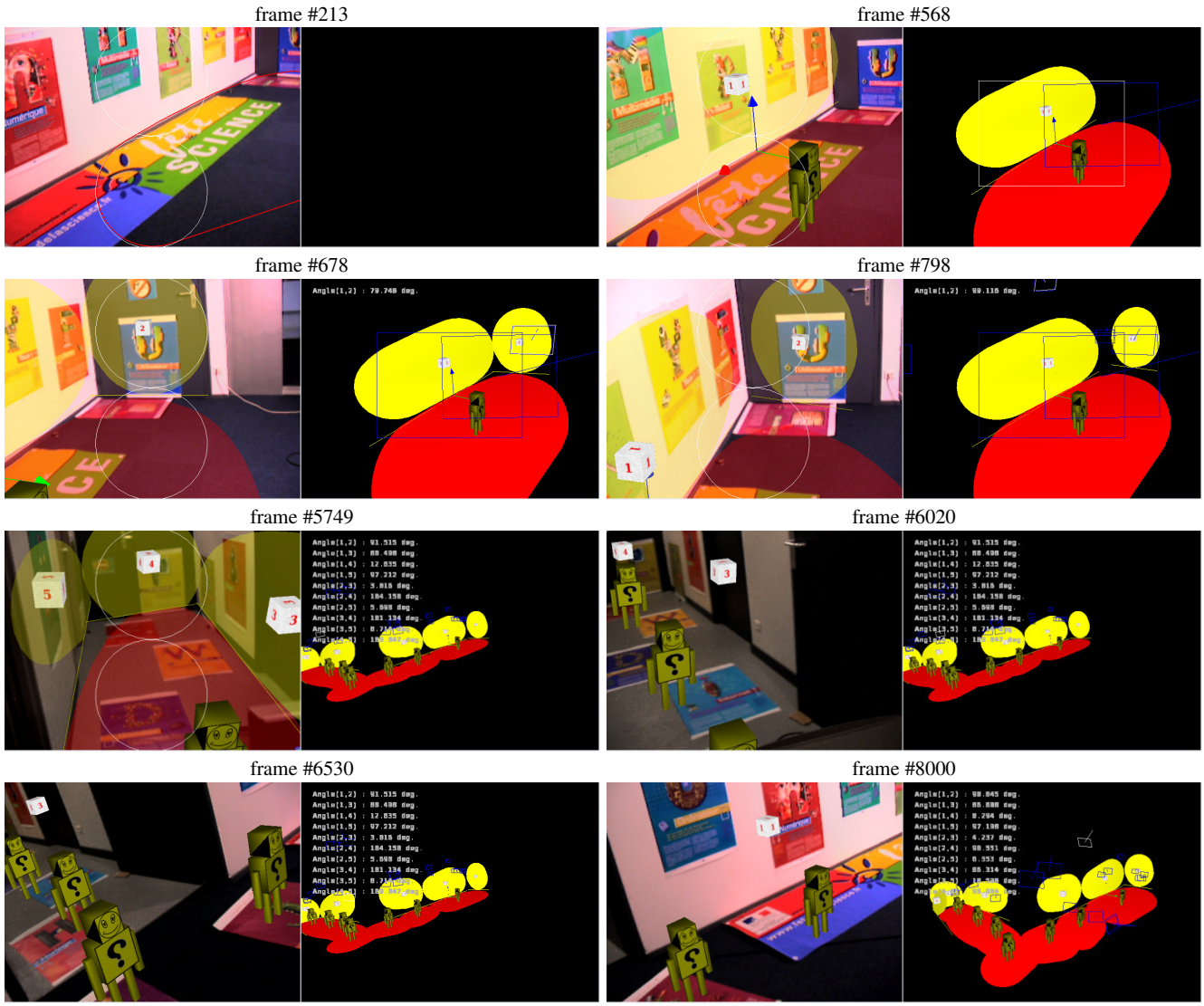


Fig. 7 On-the-fly map-building in a real indoor environment.

ing pure rotations, that were applied to interactively build the scene. Some of these motions are not favorable to automatic structure from motion. Therefore, we acquired another sequence where the camera is slowly translated and rotated so that the distance between the first and the last camera position is reasonably large (see the video on our website). We used three different methods for feature tracking:

1. Harris detection + cross-correlation matching,
2. SIFT detection and matching.
3. KLT detection and tracking [29],

Figure 9 shows two snapshots of the feature tracking sequence at the beginning and at the end of the sequence for each of the three feature tracking methods. The full sequences are available on our website. Among the three tracking methods, the first two lead to aberrant results and only the KLT method succeeds in providing consistent camera path and 3-D point cloud (fig. 10). However, even in that case, the 3-D point-cloud has lots of spurious points and no vertical plane visually appears[6].

Of course, this experiment does not prove that any automatic algorithm would fail in recovering planes in that scene, but it illustrates typical problems one may encounter. In particular, as features are rare on the walls and not very discriminant on the grass, it is difficult to track the same physical points over long periods of time. This may explain why the KLT method performed better than the two other methods: with KLT, the same points are tracked individually until tracking fails using image template similarities. In the two other cases, new sets of features are detected and matched in each frame, possibly leading to short trajecto-

[6] In order to better visualize the 3-D point cloud, a sequence captured from a virtual camera moving around it is shown on our website.

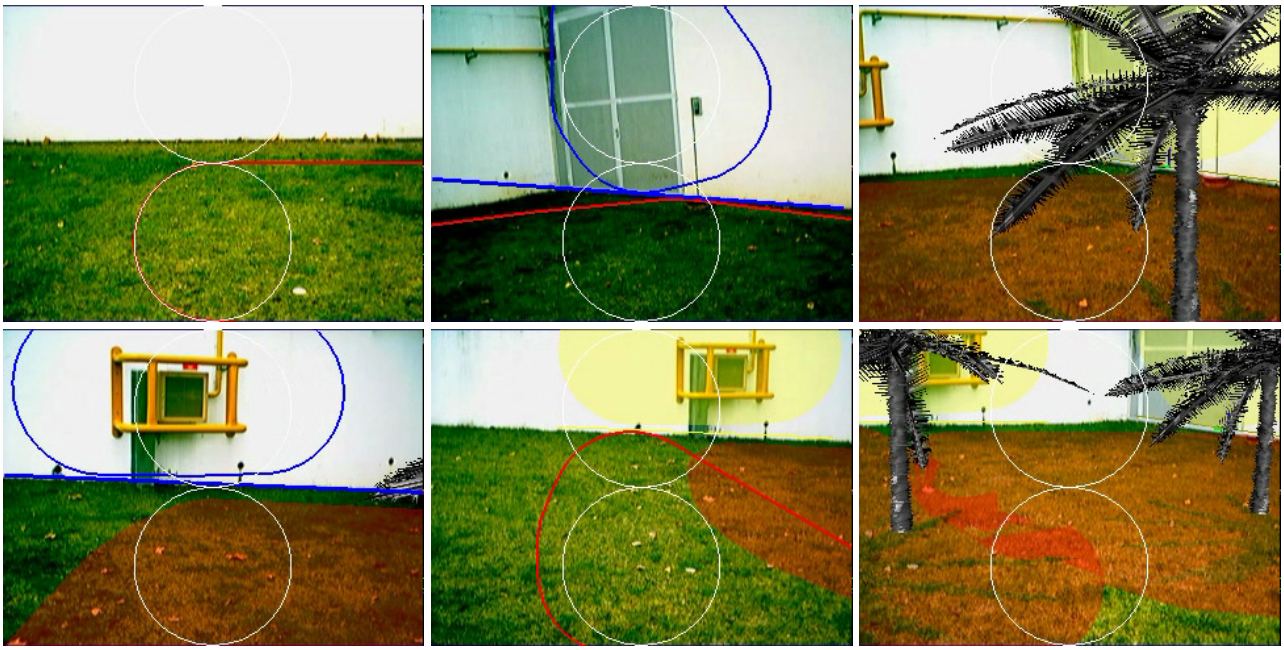


Fig. 8 Snapshots of the outdoor sequence.

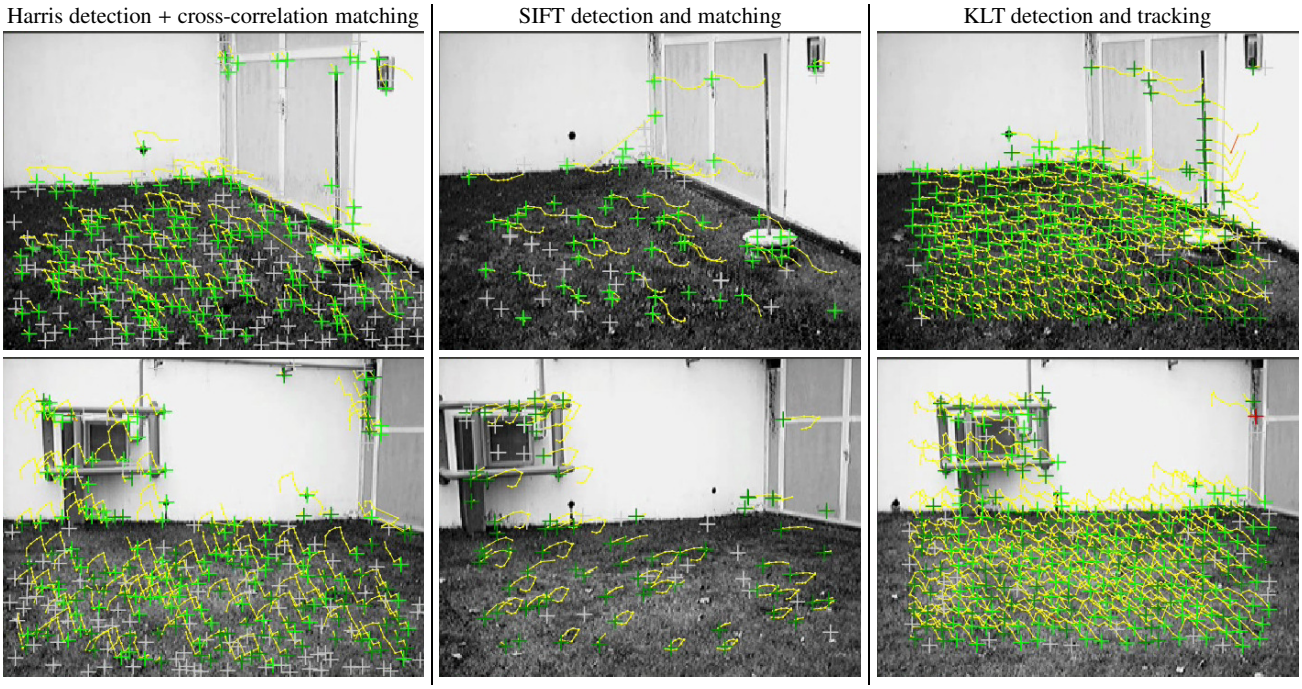


Fig. 9 Snapshots of the feature tracking sequences used for automatic structure and motion recovery (three different tracking methods). Green crosses show the tracked features, white crosses the detected but untracked features and yellow curves the feature paths in previous frames.

ries. On the contrary, 2-D blob tracking does not require that points can be tracked on long sequences. Indeed, our reconstruction procedure only depends on homographies which can be estimated from completely different sets of points matched between consecutive images.

10 Conclusion

We presented a method for interactive building of multiplanar environments that has been validated on both synthetic and real data. We have shown the benefits of using semi-automatic rather than fully automatic algorithms for online building and augmenting of multiplanar scenes. In-

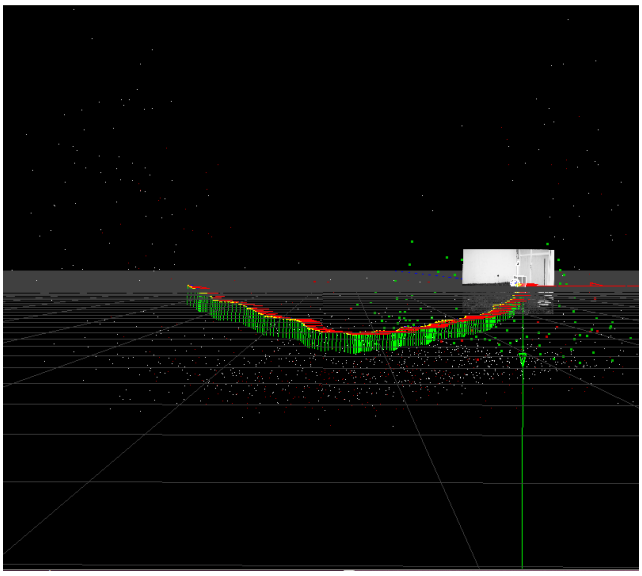


Fig. 10 3-D point-cloud and camera path obtained using KLT tracking.

teractions are intuitive to a non-expert and informative to the underlying reconstructing engine, so that only a small number of interactions are required.

Using the system in larger environments (a complete level of a building, a street, ...) would require some improvements: in order to keep a reasonable rate of exploration, the system should be allowed to perform local bundle adjustments as in [17]. The way keyframes are selected for local relocalization could also be improved: in the current implementation, keyframes are considered sequentially in arbitrary order, and the first keyframe that get enough SIFT feature correspondences is used to compute a first approximation of the camera pose. When a lot of keyframes are available, this procedure may be time consuming. Sorting the keyframes using any fast-to-compute similarity value with the current frame would make the procedure faster. A first solution is to consider highly sub-sampled images and to consider the blurred image as a descriptor as proposed in [18]. Another solution is to use sensors that may help to sort keyframes depending on their similarity with the current image position.

References

1. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for on-line non-linear/non-gaussian bayesian tracking. *IEEE Transactions on Signal Processing*, 50(2):174–188, Feb. 2002.
2. P. Bunnun and W. W. Mayol-Cuevas. OutlinAR: an assisted interactive model building system with reduced computational effort. In *IEEE / ACM International Symposium on Mixed and Augmented Reality*, pages 61–64, Los Alamitos, CA, USA, 2008. IEEE Computer Society.
3. D. Chekhlov, A. P. Gee, A. Calway, and W. Mayol-Cuevas. Ninja on a plane: Automatic discovery of physical planes for augmented reality using visual slam. In *Proceedings of the 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, ISMAR '07, pages 1–4, Washington, DC, USA, 2007. IEEE Computer Society.
4. D. Chekhlov, M. Pupilli, W. Mayol-Cuevas, and A. Calway. Real-time and robust monocular slam using predictive multi-resolution descriptors. In *2nd International Symposium on Visual Computing*, November 2006.
5. A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse. MonoSLAM: Real-time single camera SLAM. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(6):1052–1067, June 2007.
6. O. Faugeras and F. Lustman. Motion and structure from motion in a piecewise planar environment. Rapport de recherche 856, INRIA, 1988.
7. M. A. Fischler and R. C. Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. Technical Note 213, Artificial Intelligence Center, SRI International, Menlo Park, California, Mar. 1980.
8. R. Freeman and A. Steed. Interactive modelling and tracking for mixed and augmented reality. In *Proceedings of the ACM symposium on Virtual reality software and technology*, VRST '06, pages 61–64, New York, NY, USA, 2006. ACM.
9. A. P. Gee, D. Chekhlov, A. Calway, and W. W. Mayol-Cuevas. Discovering higher level structure in visual slam. *IEEE Transactions on Robotics*, 24(5):980–990, 2008.
10. C. Harris and M. Stephens. A Combined Corner and Edge Detector. In *Proceedings of 4th Alvey Conference*, Cambridge, Aug. 1988.
11. R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521623049, 2000.
12. D. N. T. Huynh, K. Raveendran, Y. Xu, K. Spreen, and B. MacIntyre. Art of defense: a collaborative handheld augmented reality board game. In *Sandbox '09: Proceedings of the 2009 ACM SIGGRAPH Symposium on Video Games*, pages 135–142, New York, NY, USA, 2009. ACM.
13. H. Jin, P. Favaro, and S. Soatto. A semi-direct approach to structure from motion. *The Visual Computer*, 19(6):377–394, 2003.
14. B. Johansson. View synthesis and 3d reconstruction of piecewise planar scenes using intersection lines between the planes. In *ICCV*, pages 54–59, 1999.
15. S. Julier, Y. Baillot, M. Lanzagorta, D. Brown, and Rosenblum". Bars: Battlefield augmented reality system. In *NATO Symposium on Information Processing Techniques for Military Systems, Istanbul, Turkey.*, pages 9–11, Oct. 2000.
16. H. Kato and M. Billinghurst. Marker Tracking and HMD Calibration for a Video-based Augmented Reality Conferencing System. In *Proceedings of the 2nd International Workshop on Augmented Reality, San Francisco*, 1999.
17. G. Klein and D. Murray. Parallel tracking and mapping for small AR workspaces. In *Proc. Sixth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'07), Nara, Japan*, November 2007.
18. G. Klein and D. Murray. Improving the Agility of Keyframe-Based SLAM. In *Proc. Europ. Conf. Computer Vision (ECCV'08)*, 2008.
19. G. Klein and D. Murray. Parallel Tracking and Mapping on a Camera Phone. In *Proc. Heighth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'09), Orlando, USA*, Oct. 2009.
20. V. Lepetit, L. Vachetti, D. Thalmann, and P. Fua. Fully Automated and Stable Registration for Augmented Reality Applications. In *Proceedings of International Symposium on Mixed and Augmented Reality, Tokyo*, pages 93–101, June 2003.

21. D. G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. Journal of Computer Vision*, 60(2):91–110, 2004.
22. N. Noury, F. Sur, and M.-O. Berger. Determining point correspondences between two views under geometric constraint and photometric consistency. Research Report RR-7246, INRIA, 04 2010.
23. J. Rabin, J. Delon, Y. Gousseau, and L. Moisan. MAC-RANSAC: a robust algorithm for the recognition of multiple objects. In *Fifth International Symposium on 3D Data Processing, Visualization and Transmission*, 2010.
24. C. Schmid and A. Zisserman. Automatic line matching across views. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 666–671, 1997.
25. G. Simon and M.-O. Berger. Detection of the Intersection Lines in Multiplanar Environments: Application to Real-Time Estimation of the Camera-Scene Geometry. In *19th International Conference on Pattern Recognition - ICPR 2008*, pages 1–4, Tampa United-States, 2008. IEEE.
26. S. Sinha, D. Steedly, and R. Szeliski. Piecewise planar stereo for image-based rendering. In *Proc. Int. Conf. Computer Vision (ICCV'09)*, pages 1881–1888, 2009.
27. T. Thormählen. Zuverlässige Schätzung der Kamerabewegung aus einer Bildfolge. Ph.D. Thesis, University of Hannover, related software 'Voodoo Camera Tracker' can be downloaded from http://www.digilab.uni-hannover.de, 2006.
28. R. Toldo and A. Fusiello. Robust multiple structures estimation with j-linkage. In *Proceedings of the 10th European Conference on Computer Vision: Part I*, ECCV '08, pages 537–547, Berlin, Heidelberg, 2008. Springer-Verlag.
29. C. Tomasi and T. Kanade. Detection and tracking of point features. Technical Report CMU-CS-91-132, Carnegie Mellon University, Apr. 1991.
30. A. van den Hengel, R. Hill, B. Ward, and A. Dick. In situ image-based modeling. *Mixed and Augmented Reality, IEEE / ACM International Symposium on*, 0:107–110, 2009.
31. F. Vigueras, M.-O. Berger, and G. Simon. Iterative multi-planar camera calibration: Improving stability using model selection. In *Vision, Video and Graphics (VVG'03), Bath, UK*, Jul 2003.
32. E. Vincent and R. Laganière. Detecting planar homographies in an image pair. In *In 2nd International Symposium on Image and Signal Processing and Analysis*, Pula, Croatia, June 2001.
33. G. Xu, J. Terai, and H. Shum. A Linear Algorithm for Camera Self-Calibration, Motion and Structure Recovery for Multi-Planar Scenes from Two Perspective Images. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, Hilton Head Island, South Carolina (USA)*, 2000.
34. L. Xu, E. Oja, and P. Kultanen. A new curve detection method: randomized Hough transform (RHT). *Pattern Recogn. Lett.*, 11:331–338, May 1990.
35. M. Zuliani, C. S. Kenney, and B. S. Manjunath. The multiransac algorithm and its application to detect planar homographies. In *In IEEE International Conference on Image Processing*, 2005.