Int J Interact Des Manuf (2010) 4:103–112
DOI 10.1007/s12008-010-0093-5

ORIGINAL PAPER

Real-time 3D features reconstruction through monocular vision

Alfredo Liverani · Francesco Leali · Marcello Pellicciari

Received: 27 June 2009 / Accepted: 9 March 2010 / Published online: 27 March 2010

Abstract A fast and interactive implementation for camera pose registration and 3D point reconstruction over a physical surface is described in this paper. The method (called SRE—Smart Reverse Engineering) extracts from a continuous image streaming, provided by a single camera moving around a real object, a point cloud and the camera's spatial trajectory. The whole per frame procedure follows three steps: camera calibration, camera registration, bundle adjustment and 3D point calculation. Camera calibration task was performed using a traditional approach based on 2-D structured pattern, while the Optical Flow approach and the Lucas-Kanade algorithm was adopted for feature detection and tracking. Camera registration problem was then solved thanks to the Essential Matrix definition. Finally a fast Bundle Adjustment was performed through the Levenberg-Marquardt algorithm to achieve the best trade-off between 3D structure and camera variations. Exploiting a PC and a commercial webcam, an experimental validation was done in order to verify precision in 3D data reconstruction and speed. Practical tests helped also to tune up several optimization parameters used to improve efficiency of most CPU time consuming algorithms, like Optical Flow and Bundle Adjustment. The method showed robust results in 3D reconstruction and very good performance in real-time applications.

A. Liverani (✉)
Engineering Faculty, DIEM, University of Bologna,
V.le Risorgimento 2, 40136 Bologna, Italy
e-mail: alfredo.liverani@unibo.it

F. Leali · M. Pellicciari
Engineering Faculty, DIMeC, University of Modena
and Reggio Emilia, Via Vignolese 905, 41125 Modena, Italy
e-mail: francesco.leali@unimore.it

M. Pellicciari
e-mail: marcello.pellicciari@unimore.it

Keywords Shape reconstruction · Reverse engineering · 3D vision · Interactive modeling · Surface remodeling

1 Introduction

1.1 Motivation and state-of-the-art

Visual navigation, entertainment, robotics and augmented reality are just few examples of industrial applications where real time camera pose estimation and computation of 3D objects location within a scene has great relevance. Until the development of ARToolkit [7] optical tracking was really not interactive and real-time applications were based on dedicated hardware equipment. Several industrial fields may exploit a real-time software capable to mix interactively real and virtual geometries in a virtual or in a real environment. So real and virtual may get closer and user-friendly. In 3D reconstruction process with optical technique, camera pose estimation is defined as the problem of determining position and orientation of pre-calibrated cameras, through the image analysis of 3D reference points, provided by external known pictures (markers) or 2D features tracked over a video sequence. Since its great influence on final performance, camera tracking appears as one of the most critical aspect, not only for Mixed Reality applications but also for many of the state of the art vision systems. In order to obtain a robust estimation of the 3D points in the scene, camera calibration has to be accurately performed (Fig. 1).

In the last years many authors have developed algorithms for camera calibration and registration. A fundamental result about camera calibration is contained in the work of Zhang [12], who described a very efficient method, based on the analysis of two or more views of a calibration pattern (usually a chessboard of known dimensions). Similar techniques

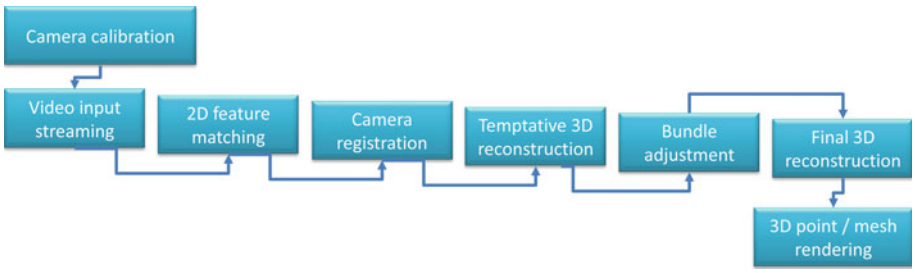

Fig. 1 From camera calibration to 3D point reconstruction

are presented by Kato [7], who focused on retrieving camera location using fiducial markers, as squares of known size with high contrast patterns in their centre, located in the 3D environment. Kutulakos [8], without metric information about the camera calibration parameters, proposed artificial markers for camera motion tracking, implementing a video-based AR system where users interactively select four or more no-coplanar points to obtain the desired values. The Structure From Motion (SFM) methodology proposes a different approach in which recovers 3D geometry from a couple of two consequent images, avoiding the use of any marker. Broida [2] developed a method based on SFM techniques using recursive algorithms that allow to estimate the position of objects in the scene, following several 2-D features in image sequence. Also [10] developed algorithms for camera position and orientation calculation starting from two shifted images. Bolles and Baker [1], as an alternative, analyzed a dense sequence of images (shot at high frequency), simulating a stereo vision with a single camera. Remaining under the assumption of a single camera use, the Optical Flow technique can be applied to recover the camera pose between two views, as it is argued in Horn [6]. Fischler and Bolles [4] developed a method which become very popular in computer vision when they introduced their RANSAC paradigm for detecting outliers from data.

In this paper the problem of interactive and monocular optical tracking, progressive 3D object reconstruction and completely marker-less statement is described and ad hoc software has been developed, overcoming the performance issue with intelligent thread based code and multiprocessing application.

1.2 Reverse Engineering and Smart Reverse Engineering

The Reverse Engineering activity is commonly considered a complex procedure that, starting from point acquisition on a physical object external surface, deals to a 3D CAD model. Conventional reverse engineering involves two main steps: the measurement of the physical object and its reconstruction as a 3D virtual object. The physical object can be measured using 3D scanning technologies such as coordinate measuring machines or computed tomography scanners, which provide outputs in the form of an unstructured point cloud, i.e. a

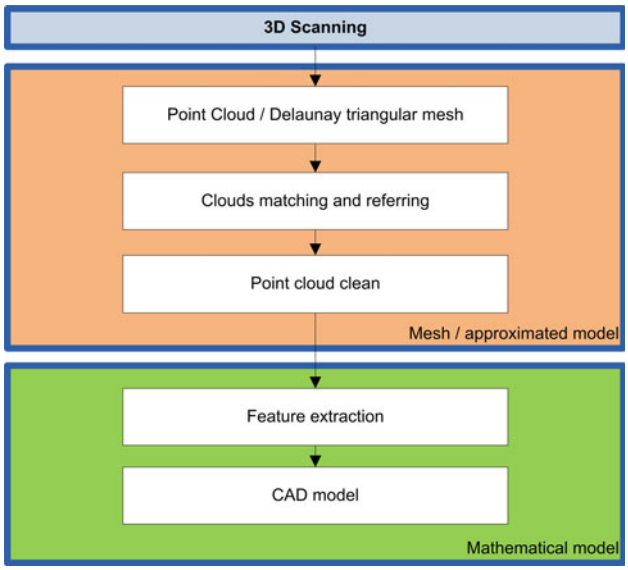

Fig. 2 From physical to virtual model: the reverse engineering steps

large set of vertices in a three dimensional coordinate system, which lacks topological information and therefore is generally not directly usable in most 3D applications. The point cloud is then usually converted to a mesh model, NURBS (Non Uniform Rational B-Spline) surface model, or CAD model through a process commonly referred to as 3D reconstruction so that it can be used for CAE and CAM. This second step of reconstruction of the virtual 3D object from the dense point cloud is an inverse problem and generally does not admit a unique solution. Most proposed approaches to reconstruction from unstructured data points build polygon meshes that interpolate or approximate the input points. In Fig. 2 a general flowchart for Reverse Engineering is represented: after the first step, where the combination of hardware and software provides the 3D point cloud, a very long time is taken by the following steps needed to integrate these very large and accurate models in a CAD system or in an interactive Virtual Reality environment. This is due to CAD, that need mathematically described surfaces and not huge meshes, and visualization and real-time systems that selected, cleaned and significantly reduced meshes.

SRE (Smart Reverse Engineering) approach is significantly different because leads to skip the data acquisition

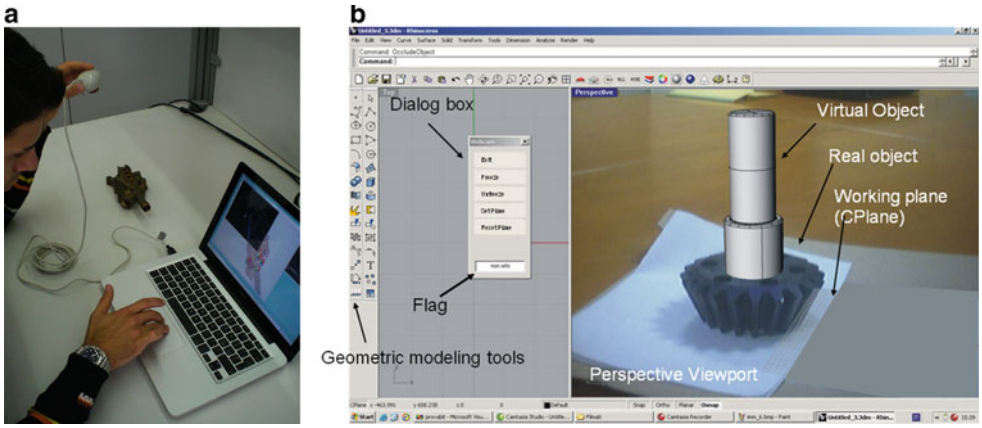

Fig. 3 **a** Camera equipment included in a CAD system for SRE. **b** SRE at work with external video on the graphics window background

stage in order to return interactively a geometry directly inside the CAD system. Furthermore SRE exploits optical low cost equipments (camera or webcam) in order to get data from physical surfaces. In Fig. 3a the camera equipment used by SRE software is showed while in Fig. 3b a SRE software prototype is working as a plug-in within Rhinoceros CAD, giving the proof that interactive reverse modelling is possible. Of course the reconstruction precision level is significantly lower than using a classic Reverse Engineering procedure, but the time needed by the whole process is reduced by over 50%.

Once introduced in a CAD environment, 3D data can be easily shared with other designers through collaborative modules already available in several commercial systems. Moreover the SRE approach may greatly help the designer in the very first project stage, where a "starting geometry" available in seconds is a good hint for the further development.

As shown in Fig. 3b, SRE reconstructs few points, refines and adds 2D features through optical flow parameters, adjusting and immediately returning them to the modelling environment. So the method is incremental: more time is spent over a surface, more 3D points may be captured. Unfortunately the optical tracking is not so accurate as, for example, laser scanning system; the final reconstruction lacks in precision, but is accurate enough to be used in a reverse modelling environment.

The paper is organized as follows. The next section treats the camera calibration process, exploited through the analysis of several image of the same calibration pattern. Section 2.1.1 is focused on the feature detection and tracking steps, based on the Optical Flow technique (Sect. 2.1.2) and the Lucas-Kanade algorithm (Sect. 2.1.3). Section 2.2 describes the epipolar geometry (Sect. 2.2.1) and the camera pose estimation task (Sect. 2.2.2). Section 2.3 and 2.4 refer to 3D reconstruction and point refinement, realized adopting the Bundle Adjustment optimization technique. The last two sections summarize the implementation details and the most important modifications introduced in algorithms for reducing the global computational cost and achieving good real time performance, validated by synthetic and real tests performed through a simple device composed by a video device and a tablet PC.

2 SRE implementation

2.1 Camera calibration

Camera calibration is crucially important for any 3D reconstruction task, because it allows to extract metric information from 2-D images. It is usually performed off-line by observing a calibration object, whose geometry and size in 3D space are known with high precision. One of the most used technique in practice requires the camera captures minimum two different orientation of a planar pattern, similar to the board shown in Fig. 4. The mathematical model of the camera, widely assumed as a pinhole, is based on the

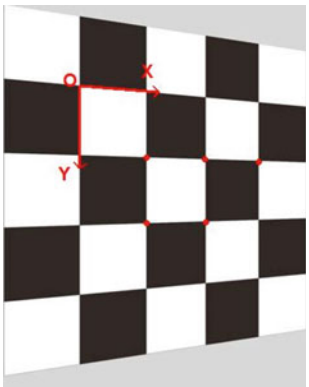

Fig. 4 2-D planar pattern used for calibration and its coordinate system

calibration matrix $\mathbf{K}$, which is 3×3 upper triangular and non-singular:

$$\mathbf{K} = \begin{pmatrix} \alpha_x & s & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{pmatrix}$$

where:

- α_x and α_y are the focal lengths along the x-axes and y-axes, measured in pixels (in general, $\alpha_y = r \cdot \alpha_x$, where r is the aspect ratio of pixels, but it is usually equal to 1);
- $\mathbf{C_0} = [x_0, y_0]$ is the principal point, which is the projection center on the image plane, obtained as the intersection between the image plane and the line perpendicular to it and passing through the camera center;
- $s = \tan\theta$, where θ is the angle between the x and y axes of the image, is the skew coefficient (the images axes are generally perpendiculars so $\theta = 90°$).

External parameters depend on position and orientation of the real camera with respect to the world reference system. The six external parameters are computed starting from the $\mathbf{t}$ vector and the $\mathbf{R}$ matrix:

- three parameters for the translation vector $\mathbf{t} = \left[t_x, t_y, t_z\right]$;
- three parameters for the rotation matrix

$$\mathbf{R} = [\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3]$$

with:

$$\mathbf{r}_1 = \begin{pmatrix} \cos(\beta) \cdot \cos(\gamma) \\ -\cos(\beta) \cdot \sin(\gamma) \\ \sin(\beta) \end{pmatrix}$$

$$\mathbf{r}_2 = \begin{pmatrix} \sin(\alpha) \cdot \sin(\beta) \cdot \cos(\gamma) + \cos(\alpha) \cdot \sin(\gamma) \\ -\sin(\alpha) \cdot \sin(\beta) \cdot \sin(\gamma) + \cos(\alpha) \cdot \cos(\gamma) \\ -\sin(\alpha) \cdot \cos(\beta) \end{pmatrix}$$

$$\mathbf{r}_3 = \begin{pmatrix} -\cos(\alpha) \cdot \sin(\beta) \cdot \cos(\gamma) + \sin(\alpha) \cdot \sin(\gamma) \\ \cos(\alpha) \cdot \sin(\beta) \cdot \sin(\gamma) + \sin(\alpha) \cdot \cos(\gamma) \\ \cos(\alpha) \cdot \cos(\beta) \end{pmatrix}$$

During the calibration stage a printed marker is projected and then acquired by the camera. Through the analysis of several images of this pattern, at least n corresponding points $(\mathbf{x}_i, \mathbf{X}_i)$, $i = 1, \ldots, n$, need to be provided, where $\mathbf{X}_i$ is a point on the scene and $\mathbf{x}_i$ is its image on the projection plane.

According to [12], the calibration algorithm for the camera is composed of three fundamental parts:

1. computing the homography between the model plane and its image, for each image considered;
2. imposing the homography constraints and solve an homogeneous linear system to find the matrix $\mathbf{K}$;
3. computing matrix $\mathbf{R}$ and vector $\mathbf{t}$.

The model plane is defined as the plane where the calibration pattern lies, with respect to the world reference system. Without loss of generality, the model plane can be assumed as $Z = 0$. If we call $\mathbf{P}$ the projection matrix $\mathbf{P} = \mathbf{KR}\left[\mathbf{I} \,|{-}\mathbf{t}\right]$, $\mathbf{I}$ is the 3×3 Identity matrix, the following equation can be written:

$$\lambda \mathbf{x_i} = \mathbf{P}\mathbf{X}_i = \mathbf{KR}\left[\mathbf{I} \,|{-}\mathbf{t}\right]\mathbf{X}_i \tag{1}$$

Such relation is valid for each $\mathbf{x}_i$ in the image plane and for each $\mathbf{X}_i$ in the 3D scene, where $\lambda \in \mathbf{R}$ is an arbitrary scale factor. Writing Eq. 1 in homogeneous coordinate:

$$\lambda \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} = \mathbf{K}\left[\mathbf{R} \,|\mathbf{R}\,(-\mathbf{t})\right] = \mathbf{K}\begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{r}_3 & |\tilde{\mathbf{t}}| \end{bmatrix} \begin{bmatrix} X_i \\ Y_i \\ 0 \\ 1 \end{bmatrix}$$

$$\mathbf{K}\left[\mathbf{R} \,|\mathbf{R}\,(-\mathbf{t})\right] = \mathbf{K}\begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & |\tilde{\mathbf{t}}| \end{bmatrix} \begin{bmatrix} X_i \\ Y_i \\ 1 \end{bmatrix}$$

with $\tilde{t} = \mathbf{R}\,(-\mathbf{t})$. Denoting with $\tilde{X}_i = [X_i, Y_i, 1]^T$ an homogeneous point in the plane $Z = 0$, point $\tilde{X}_i$ is related to its image $\tilde{x}_i = [x_i, y_i, 1]^T$ by the following equations:

$$\lambda \tilde{\mathbf{x}}_i = \mathbf{H}\tilde{X}_i \quad \mathbf{H} = \mathbf{K}\begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & |\tilde{\mathbf{t}}| \end{bmatrix} \tag{2}$$

Matrix $\mathbf{H}$ is the 3×3 Homography matrix between the model plane and the image plane:

$$\mathbf{H} = \mathbf{K}\begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & |\tilde{\mathbf{t}}| \end{bmatrix} = [\mathbf{h}_1 \quad \mathbf{h}_2 \quad \mathbf{h}_3]$$

The columns of a rotation matrix are orthonormal, so [12]:

$$\mathbf{h}_1^T \mathbf{K}^{-T} \mathbf{K}^{-1} \mathbf{h}_2 = 0 \tag{3}$$

and

$$\mathbf{h}_1^T \mathbf{K}^{-T} \mathbf{K}^{-1} \mathbf{h}_2 = \mathbf{h}_2^T \mathbf{K}^{-T} \mathbf{K}^{-1} \mathbf{h}_2 \tag{4}$$

The following 3×3 symmetric matrix, called $\mathbf{B}$,

$$\mathbf{B} = \mathbf{K}^{-T}\mathbf{K}^{-1}$$

can be defined by a 6-dimensional vector $\mathbf{b} = [B_{11}, B_{12}, B_{22}, B_{13}, B_{23}, B_{33}]^T$. Denoting the i-th column of matrix $\mathbf{H}$ as $\mathbf{h}_i = [h_{i1}, h_{i2}, h_{i3}]^T$; the following relation is valid:

$$\mathbf{h}_i^T \mathbf{B} \mathbf{h}_j = \mathbf{v}_{ij}^T \mathbf{B} \tag{5}$$

where

$$\mathbf{v}_{ij} = \big[h_{i1}h_{j1}, h_{i1}h_{j2} + h_{i2}h_{j1}, h_{i2}h_{j2}, h_{i3}h_{j1} + h_{i1}h_{j3}, \ldots, h_{i3}h_{j2} + h_{i2}h_{j3}, h_{i3}h_{j3}\big]^T$$

Equations 3 and Eq. 4 can be rewritten as:

$$\begin{bmatrix} \mathbf{v}_{12}^T \\ (\mathbf{v}_{11} - \mathbf{v}_{22})^T \end{bmatrix} \mathbf{b} = 0 \tag{6}$$

Considering n images of the model plane, written as in (6) the following system can be obtained:

$$\mathbf{V}\mathbf{b} = 0$$

where $\mathbf{V}$ is a $2n \times 6$ matrix. This system can be solved applying the Singular Value Decomposition Theory. Once $\mathbf{b}$ is estimated, camera intrinsic matrix $\mathbf{K}$ can be computed. See Zhang [12] for a more detailed algorithm description. Once $\mathbf{K}$ is known, also the camera extrinsic parameters for each image are easily computed:

$$\mathbf{r}_1 = \lambda \mathbf{K}^{-1}\mathbf{h}_1$$
$$\mathbf{r}_2 = \lambda \mathbf{K}^{-1}\mathbf{h}_2$$
$$\mathbf{r}_3 = \mathbf{r}_1 \times \mathbf{r}_2$$
$$\mathbf{t} = \lambda \mathbf{K}^{-1}\mathbf{h}_3$$

with $\lambda = \frac{1}{\|\mathbf{K}^{-1}\mathbf{h}_1\|}$. Applying the Maximum Likelyhood Estimate algorithm, matrices $\mathbf{K}$, $\mathbf{R}$ and vector $\mathbf{t}$ can be refined and the desired precision can be obtained.

2.1.1 Feature detection and extraction

Feature detection and feature correspondence are widely considered as the key steps for an accurate 3D reconstruction. Such features, also called points of interest, can be defined as locations characterized by large intensity variations in every direction [9]. Two points, belonging to two different images $\mathbf{I}_{j-1}$ and $\mathbf{I}_j$, are conjugated if they are the projection of the same 3D point in the scene. These conjugated points can be found using the corner method. According to this method, a point on the image is classified in dependence of its directional variation:

- it is a plane point if there is no variation;
- it is an edge point if there is a variation along one direction;
- it is a corner point if the variations are along all directions.

Given the point $\mathbf{P}(u, v)$ on the first image and its neighbourhood $\mathbf{W}$ and $\mathbf{d} \in \mathbf{W}$, then the following function can be defined:

$$\mathbf{E}_h(\mathbf{P}) = \sum_{\mathbf{d}\in\mathbf{W}} [\mathbf{I}(\mathbf{P}+\mathbf{d}) - \mathbf{I}(\mathbf{P}+\mathbf{d}+\mathbf{h})]^2 \tag{7}$$

where $\mathbf{I}$ represents the image intensity. Function $\mathbf{E}_h$ computes the variation of brightness between two points of $\mathbf{W}$ for a displacement $\mathbf{h}$, where $\mathbf{h}$ is the maximum limit of the computable disparity. Using truncated Taylor's series, $\mathbf{E}_h(\mathbf{P})$ has the following form:

$$\begin{aligned}\mathbf{E}_h(\mathbf{P}) &= \sum_{\mathbf{d}\in\mathbf{W}} \left[\nabla \mathbf{I}(\mathbf{P}+\mathbf{d})^T \mathbf{h}\right]^2 \\ &= \sum_{\mathbf{d}\in\mathbf{W}} \mathbf{h}^T (\nabla \mathbf{I}(\mathbf{P}+\mathbf{d})) (\nabla \mathbf{I}(\mathbf{P}+\mathbf{d}))^T \mathbf{h} \\ &= \sum_{\mathbf{d}\in\mathbf{W}} \mathbf{h}^T \begin{pmatrix} \mathbf{I}_u^2 & \mathbf{I}_u\mathbf{I}_v \\ \mathbf{I}_u\mathbf{I}_v & \mathbf{I}_v^2 \end{pmatrix} \mathbf{h}\end{aligned}$$

where $\nabla \mathbf{I}(\mathbf{P}+\mathbf{d}) = [\mathbf{I}_u \mathbf{I}_v]^2$. A Gaussian weight function $w(*)$ can be introduced on $\mathbf{W}$ in order to smooth the result and to limit the affect of noise in the window; w has the expression:

$$w(\mathbf{d}) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-\|\mathbf{d}\|^2}{2\sigma^2}}$$

The Taylor's series can be now rewritten as:

$$\mathbf{E}_h(\mathbf{P}) = \mathbf{h}^T \begin{pmatrix} \sum\limits_{\mathbf{d}\in\mathbf{W}} \mathbf{I}_u^2 w(\mathbf{d}) & \sum\limits_{\mathbf{d}\in\mathbf{W}} \mathbf{I}_u\mathbf{I}_v w(\mathbf{d}) \\ \sum\limits_{\mathbf{d}\in\mathbf{W}} \mathbf{I}_u\mathbf{I}_v w(\mathbf{d}) & \sum\limits_{\mathbf{d}\in\mathbf{W}} \mathbf{I}_v^2 w(\mathbf{d}) \end{pmatrix} \mathbf{h} \tag{8}$$

Considering the following pseudo-hessian 2×2 matrix $\mathbf{M}$:

$$\mathbf{M} = \begin{pmatrix} \sum\limits_{\mathbf{d}\in\mathbf{W}} \mathbf{I}_u^2 w(\mathbf{d}) & \sum\limits_{\mathbf{d}\in\mathbf{W}} \mathbf{I}_u\mathbf{I}_v w(\mathbf{d}) \\ \sum\limits_{\mathbf{d}\in\mathbf{W}} \mathbf{I}_u\mathbf{I}_v w(\mathbf{d}) & \sum\limits_{\mathbf{d}\in\mathbf{W}} \mathbf{I}_v^2 w(\mathbf{d}) \end{pmatrix} \tag{9}$$

Equation 8 can be rewritten in the following form:

$$\mathbf{E}_h(\mathbf{P}) = \mathbf{h}^T \mathbf{M} \mathbf{h}$$

$\mathbf{M}$ is symmetric and positive-defined, so it is diagonalizable and its eigenvalues are not-negative. In order to classify the point $\mathbf{P}$, the eigenvalues λ_1, λ_2 of $\mathbf{M}$ (for unitary displacements $\mathbf{h}$) have to be computed:

- if $\lambda_1 \approx \lambda_2 \approx 0$, $\mathbf{P}$ is a plane point (the considered window region has approximately a constant intensity);
- if $\lambda_1 \approx 0$ and $\lambda_2 > 0$, $\mathbf{P}$ is an edge point (only shifts along the edge cause little change in E);
- if $\lambda_1 > 0$ and $\lambda_2 > 0$, $\mathbf{P}$ is a corner point (shifts in any direction will increase $\mathbf{E}$).

Let $\lambda_1 > \lambda_2$: according to this classification, $\mathbf{P}$ is a corner if:

$$\lambda_1 > 0 \quad \frac{\lambda_2}{\lambda_1} \to 1.$$

Thanks to such approach a good number of feature points can be located on the first image with high accuracy.

2.1.2 Feature matching by optical flow

After having determined a sufficient number of points of interest on the first image, the same points have to be

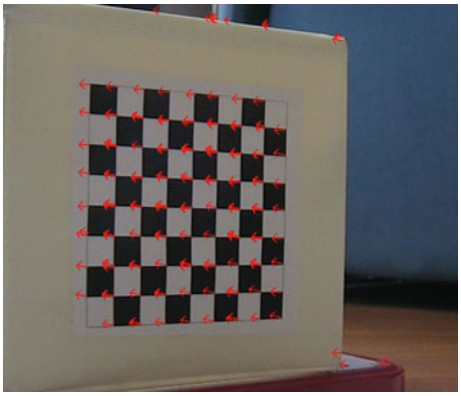

Fig. 5 Finding the optical flow vector

found on the second frame, through a process called matching of corresponding points. Seven pairs of corresponding points minimum are needed to correctly determine the geometric relationship between two images. Image matching represents a fundamental aspect of many problems in computer vision, including object or scene recognition, 3D reconstruction from multiple images, stereo correspondence and motion tracking. Feature extraction on different images can be done applying the concept of Optical Flow. The expression "Optical Flow" denotes a vector field defined on the image plane and obtained following the motion of some points of interest in the scene (Fig. 5), due to a relative motion of the objects with respect to the viewer. The Optical Flow method is based on the widely accepted assumption that points laying at the same location (confirmed by corresponding pixel values) maintain their brightness constant over time. Defined as $\mathbf{I}(x, y)$ and $\mathbf{J}(x, y)$ the greyscale values of two images at the location $\mathbf{x} = [x, y]^T$, x and y are the two pixel coordinates of a generic image point $\mathbf{p}$. Considering an image point $\mathbf{p} = \left[p_x, p_y\right]^T$ on the first image $\mathbf{x} = [x, y]^T$, the feature tracking goal is finding the location $\mathbf{q} = \mathbf{p} + \mathbf{v} = \left[p_x + v_x, p_y + v_y\right]^T$ on the second image $\mathbf{J}$ such as $\mathbf{I}(\mathbf{p})$ and $\mathbf{J}(\mathbf{q})$ show similarities. The vector $\mathbf{v} = \left[v_x, v_y\right]^T$ is the image velocity at $\mathbf{p}$, also known as the Optical Flow at $\mathbf{p}$.

2.1.3 Lucas-Kanade algorithm for feature tracking

One of the most efficient feature tracking algorithms is the Lukas-Kanade differential algorithm. Such approach addresses the local Taylor series approximation for the image signal to calculate the motion between two frames. The assumption of this method is that the image intensity remains constant for small shifting displacements:

$$\mathbf{I}(x, y, t) = \mathbf{I}(x + \delta x, y + \delta y, t + \delta t) \tag{10}$$

where $\mathbf{I}(x, y, t)$ is the image intensity at point (x, y) at time t.

Assuming the movement as small enough to avoid spatial discontinuities in reflectance, the image constraint at time $t + \delta t$ can be developed, with Taylor series, to get:

$$\mathbf{I}(x + \delta x, y + \delta y, t + \delta t) = \mathbf{I}(x, y, t) + \frac{\partial \mathbf{I}}{\partial x}\delta x + \frac{\partial \mathbf{I}}{\partial y}\delta y + \frac{\partial \mathbf{I}}{\partial t} + o\left(t^2\right)$$

After the substitution in Eq. (10), the constraint equation becomes:

$$\frac{d\mathbf{I}(x, y, t)}{dt} = o\left(t^2\right)$$

Or

$$\mathbf{I}_x(x, y)\, v_x + \mathbf{I}_y(x, y)\, v_y + \mathbf{I}_t(x, y) = 0 \tag{11}$$

The "brightness change constraint equation" expressed by Eq. (11) is a linear constraint on the image velocity components v_x and v_y with the partial derivatives of brightness, $\mathbf{I}_x$, $\mathbf{I}_y$ and $\mathbf{I}_t$ as coefficients. Equation (11) is equivalent to

$$\mathbf{I}_x(x, y)\, v_x + \mathbf{I}_y(x, y)\, v_y = -\mathbf{I}_t(x, y) \tag{12}$$

Equation (12) shows two unknowns and cannot be immediately solved without additional constraints. This is known as the aperture problem of the Optical Flow algorithms. Another set of equations, given by some additional constraint, is needed to solve the Optical Flow ambiguity. The solution given by Lucas and Kanade in is a non-iterative method which assumes that the flow is locally constant (brightness smoothness constraint). Using the Optical Flow equation for a set of adjacent pixels and assuming that all of them have the same velocity, the Optical Flow computation task can be reduced to solve a linear system and the approximate solution is found using the least squares method. Adding the brightness smoothness constraint, the following system can be obtained:

$$\begin{bmatrix} I_{x1} & I_{y1} \\ I_{x2} & I_{y2} \\ \cdots & \cdots \\ I_{xn} & I_{yn} \end{bmatrix} \begin{bmatrix} v_x \\ v_y \end{bmatrix} = \begin{bmatrix} -I_{t1} \\ -I_{t2} \\ \cdots \\ -I_{tn} \end{bmatrix} \tag{13}$$

valid for n pixels in the considered neighbourhood of point (x, y).

System (13) can be rewritten in a more compact form:

$$\mathbf{A}\mathbf{v} = \mathbf{b} \tag{14}$$

Applying the least squares method a solution of (14) is:

$$\mathbf{v} = \left(\mathbf{A}^T\mathbf{A}\right)^{-1}\mathbf{A}^T\mathbf{b} \tag{15}$$

The Optical Flow estimated in (15) is then interpolated and refined in order to achieve a correct velocity vector for the point movement. A Gaussian weighting function $w(i, j)$, where $i, j \in [1, \ldots, m]$ and m is the window size in two

dimensions, should be added to give more prominence to the center pixel of the window. The iteration can be stopped when the computed pixel residual value is smaller then a given threshold (usually fixed at 0.03 pixels) or when a maximum number of iterations is reached (usually 20 iterations). The Lukas-Kanade iterative method provides sufficient local tracking accuracy.

2.2 Camera registration

Section 3 described the method used to obtain feature extraction and correspondence between given images. If the process can be leaded on images acquired from different directions and for a sufficent number of points and lines, then camera pose during its motion may be calculated.

2.2.1 Epipolar geometry

Realized a good correspondence of features, the geometric constrains between different views can be established, as described by the Epipolar Geometry theory for two images of a single scene (Fig. 6). Such images can be taken by different cameras or by a moving single camera at different time. However, when the motion between the two images is not known, the epipolar geometry is also undetermined. During the 1990s, the concept of Fundamental Matrix **F** was introduced by Faugeras [3], in order to generalize Epipolar Geometry for uncalibrated cameras.

Given two images $\mathbf{I}_j$ and $\mathbf{I}_{j+1}$, taken from the same camera in different positions, the fundamental 3×3 matrix **F** establishes the relation between an epipolar line on $\mathbf{I}_{j+1}$ and the corresponding point on $\mathbf{I}_j$.

Each camera position is described by its centre (O, O') and the relative image plane. The epipolar plane contains the 3D object point **P** and the camera centres of projection. Epipoles (e, e') represent the intersection between the image plan and the baseline connecting O and O' and laying on the epipolar plane. Epipolar lines are defined as the intersection between the epipolar plane and the image plane.

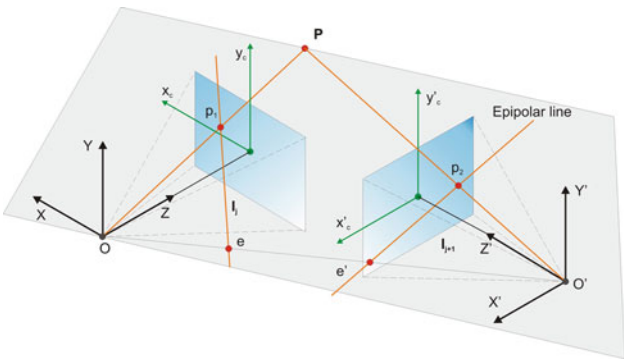

Fig. 6 Epipolar geometry

After introducing the 3×3 skew matrix **N** associated to the vector **t**:

$$\mathbf{N} = \begin{pmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{pmatrix}$$

the Fundamental Matrix has the following expression, (the scale factor is not included):

$$\mathbf{F} \cong \mathbf{K}^{-T}\mathbf{N}\mathbf{R}\mathbf{K}^{-1} \tag{16}$$

where **R** and **t** are the relative rotation and translation between the two positions of the moving camera. If n pairs of corresponding points $(\mathbf{x}, \mathbf{x}')$ in the two images are known, the following epipolar equation is valid:

$$\mathbf{x}'^{T}\mathbf{F}\mathbf{x} = 0$$

The epipoles **e** and **e′**, respectively of $\mathbf{I}_j$ and $\mathbf{I}_{j+1}$, are the eigenvectors of **F** corresponding to the eigenvalue zero:

$$\mathbf{F}\mathbf{e} = 0 \qquad \mathbf{F}\mathbf{e}' = 0$$

Appling the epipolar equation to all the n corresponding points, an homogeneous system, whose solution is the matrix **F**, is obtained. This system can be rewritten in a more compact form:

$$\mathbf{B}\mathbf{X} = 0 \tag{17}$$

for an appropriate matrix **B** and it is usually an overdeterminated system. Such system can be solved exploiting the Singular Value Decomposition, so **F** can be determined. It is important to normalize the image coordinates before solving the linear equations previously mentioned. Normalization avoids the columns of matrix **B** in Eq. (17) differing even several orders of magnitude and avoids error concentrating on the coefficients corresponding to the smaller columns. Some important properties of the matrix **F** for the pair of positions $(\mathbf{C}, \mathbf{C}')$ are that $\mathbf{F}^T$ is the fundamental matrix for $(\mathbf{C}', \mathbf{C})$ and the matrix is not of full rank, so $\det(\mathbf{F}) = 0$.

2.2.2 Camera pose estimation

Given the matrix **K** and the Fundamental Matrix **F** for a couple of subsequent images, camera pose during the real videocamera movement can be interactively extracted. Without loss of generality, the world reference system is supposed to coincide with the first camera reference system and, under this assumption, the method presented here guarantees an efficient computation of the rotation and translation of the second camera position with respect to the first one. This movement is expressed by a 3×3 rotation matrix **R** and a 1×3 translation vector **t**. Hartley [5] analyzes the properties of the Essential Matrix **E**, which is a 3×3 matrix derived

from the Fundamental Matrix **F**:

$$\mathbf{E} = \mathbf{K}^T \mathbf{F} \mathbf{K} = \mathbf{N}\mathbf{R}$$

As the fundamental matrix, the essential matrix **E** is not of full rank (det (**E**) = 0) and it can be used to solve the registration problem through the Singular Value Decomposition method. Thanks to this decomposition, **E** can be rewritten as

$$\mathbf{E} = \mathbf{U}\mathbf{S}\mathbf{V}^T$$

where **U**, **V** are orthogonal 3×3 matrices ($\mathbf{U}\mathbf{U}^T = \mathbf{V}\mathbf{V}^T = \mathbf{I}$) and **S** is a 3×3 diagonal matrix whose diagonal elements are the singular values of matrix **E**. Let **W** be the matrix:

$$\mathbf{W} = \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

The algorithm now finds two possible rotation matrices $\mathbf{R}_1$ and $\mathbf{R}_2$ between the two positions:

$$\mathbf{R}_1 = \mathbf{U}\mathbf{W}\mathbf{V}^T$$
$$\mathbf{R}_2 = \mathbf{U}\mathbf{W}^T\mathbf{V}^T$$

and two possible translation vectors:

$$\mathbf{t}_1 = \mathbf{U}_3 \quad \mathbf{t}_2 = -\mathbf{U}_3$$

where $\mathbf{U}_3$ is the last column of matrix **U**. The difference in sign of the two possible translation vectors is due to the fact that t can have opposite direction (from the first camera to the second one and vice versa) and the two rotation matrices differs of a rotation of 180 degrees around the baseline. This ambiguity, called projective ambiguity, can be immediately eliminated finding the unique transformation which maps 2-D points in front of both positions and for which all the 3D points have positive z-values.

2.3 3D reconstruction

Obtained the values of the matrices **K** and **F**, the camera location at the j-th frame is well-known. Exploiting an inverse projection computation, keeping the world coordinate system coinciding with the system of the first camera, the most visual feature 3D coordinates can be calculated. The projection matrix $\mathbf{P}_{j-1}$, which maps an image point $\mathbf{x}_i$ from the first image to a 3D point $\mathbf{X}_i$, has the following form:

$$\mathbf{P}_{j-1} = \mathbf{K}\,[\mathbf{I}\,|\mathbf{0}\,] \quad \Rightarrow \quad \mathbf{x}_i = \mathbf{P}_{j-1}\mathbf{X}_i \tag{18}$$

The projection matrix $\mathbf{P}_j$ analogously transforms an image point $\mathbf{x}'_i$ from the second image to a 3D point $\mathbf{X}_i$:

$$\mathbf{P}_j = \mathbf{K}\mathbf{R}\,[\mathbf{I}\,|-\mathbf{t}\,] \quad \Rightarrow \quad \mathbf{x}'_i = \mathbf{P}_j\mathbf{X}_i \tag{19}$$

Having already determined $\mathbf{P}_{j-1}$ and $\mathbf{P}_j$, the coordinates of the point **X** corresponding to the two projection points considered may be calculated. Define now

$$\mathbf{P}_{j-1} = [\mathbf{c}_1 \quad \mathbf{c}_2 \quad \mathbf{c}_3]^T \; \mathbf{P}_j = \left[\mathbf{c}'_1 \quad \mathbf{c}'_2 \quad \mathbf{c}'_3\right]^T$$

Equation (18) can be rewritten as:

$$\omega\,[x_i \quad y_i \quad 1]^T = [\mathbf{c}_1 \quad \mathbf{c}_2 \quad \mathbf{c}_3]^T\,\mathbf{X}_i \tag{20}$$

where ω stands for the homogeneous coordinate, and Eq. (19) becomes:

$$\omega\,[x_i \quad y_i \quad 1]^T = [\mathbf{c}_1 \quad \mathbf{c}_2 \quad \mathbf{c}_3]^T\,\mathbf{X}_i \tag{21}$$

Developing Eq. (20), a system of three equations is obtained:

$$\begin{cases} \omega x_i = \mathbf{c}_1\mathbf{X}_i \\ \omega y_i = \mathbf{c}_2\mathbf{X}_i \\ \omega = \mathbf{c}_3\mathbf{X}_i \end{cases}$$

and it can be reduced to

$$\begin{cases} \mathbf{c}_3\mathbf{X}_i x_i = \mathbf{c}_1\mathbf{X}_i \\ \mathbf{c}_3\mathbf{X}_i y_i = \mathbf{c}_2\mathbf{X}_i \end{cases} \Rightarrow \begin{cases} \mathbf{c}_3\mathbf{X}_i x_i - \mathbf{c}_1\mathbf{X}_i = 0 \\ \mathbf{c}_3\mathbf{X}_i y_i - \mathbf{c}_2\mathbf{X}_i = 0 \end{cases} \Rightarrow \begin{bmatrix} \mathbf{c}_3 x_i - \mathbf{c}_1 \\ \mathbf{c}_3 y_i - \mathbf{c}_2 \end{bmatrix} \mathbf{X}_i = 0 \tag{22}$$

In the same way, starting from Eq. (21) the final system is:

$$\begin{bmatrix} \mathbf{c}'_3 x'_i - \mathbf{c}'_1 \\ \mathbf{c}'_3 y'_i - \mathbf{c}'_2 \end{bmatrix} \mathbf{X}_i = 0 \tag{23}$$

Combining together the relations (22) and (23), the final triangulation equation is:

$$\begin{bmatrix} \mathbf{c}_3 x_i - \mathbf{c}_1 \\ \mathbf{c}_3 y_i - \mathbf{c}_2 \\ \mathbf{c}'_3 x'_i - \mathbf{c}'_1 \\ \mathbf{c}'_3 y'_i - \mathbf{c}'_2 \end{bmatrix} \mathbf{X}_i = 0 \tag{24}$$

which appears as a linear homogeneous system and it can be solved using the Least Squares Method. For each pair of corresponding points on the two images, a 3D point **X** expressed in the world reference system can now be determined. The unknown scale factor in Eq. (16) is usually determined using a known length on the scene and observing its variation.

2.4 Bundle adjustment and point refinement

Bundle Adjustment was originally conceived for photogrammetry and then widely adopted in computer vision algorithms. This technique aims to minimize the re-projection error between observed and predicted image points, which is expressed as sum of squares of a number of non-linear real-valued functions. The Levenberg-Marquardt (LM) is widely used to solve Bundle Adjustment task.

2.4.1 The Levenberg-Marquardt algorithm

The LM algorithm is an iterative technique that locates the minimum of a multivariate function, expressed as the sum of squares of non-linear real-valued functions, and it is essentially a combination of steepest descent and the Gauss-Newton method. The LM algorithm is more robust than the Gauss-Newton algorithm since it can find a good solution even starting very far from the final minimum. During the last decade, this method has been widely adopted in the field of Computer Vision because of its low computational cost with respect to many other algorithms.

2.4.2 The bundle adjustment algorithm

Consider n 3D points from m different views, where an image point is denoted by $\mathbf{x}_{ij}$, which is the projection of the i-th point on image j. The BA algorithm is used to refine a set of initial camera and structure parameters estimated for finding the final values that best predict the locations of the observed n points in the m images considered. Each camera j is parameterized by a vector $\mathbf{c}_j$ and each 3D point i by a vector $\mathbf{v}_i$.

According to the previous notation, Bundle Adjustment minimizes the re-projection error with respect to all 3D points and camera parameters, expressed by the following function:

$$\min_{\mathbf{c}_j,\mathbf{v}_i} \sum_{i=0}^{n} \sum_{j=0}^{m} d_E \left(\mathbf{P}\left(\mathbf{c}_j, \mathbf{v}_i\right), \mathbf{x}_{i,j}\right)^2$$

where $\mathbf{P}\left(\mathbf{c}_j, \mathbf{v}_i\right)$ is the predicted projection of point i on image j end d_E is the Euclidean distance in $\mathbf{R}^2$. The camera parameters contained in vector $\mathbf{c}_j$ are expressed by a translation vector $\mathbf{t}_j = \left[t_x^j, t_y^j, t_z^j\right]$ and a quaternion $\mathbf{q}_j = \left[\cos\frac{\theta_j}{2}, u_1^j \sin\frac{\theta_j}{2}, u_2^j \sin\frac{\theta_j}{2}, u_3^j \sin\frac{\theta_j}{2}\right]$, which expresses the rotation of angle θ_j and axes $\mathbf{u}_j = \left[u_1^j, u_2^j, u_3^j\right]$. Let $\mathbf{B}_0$ be an initial parameter estimate, defined by all parameters describing the m projection matrices and the n 3D points

$$\mathbf{B}_0 = \left(\mathbf{c}_1^T, \ldots, \mathbf{c}_m^T, \mathbf{v}_1^T, \ldots, \mathbf{v}_n^T\right)$$

The corresponding measurement vector $\mathbf{M}_0$, made by the 2D image points, is defined by a functional relation $\mathbf{M}_0 = f\left(\mathbf{B}_0\right)$ and has the following form:

$$\mathbf{M}_0 = \left(\mathbf{x}_{11}^T, \ldots, \mathbf{x}_{1m}^T, \ldots, \mathbf{x}_{n1}^T, \ldots, \mathbf{x}_{nm}^T\right)$$

If $\sum_M$ denotes the covariance matrix corresponding to the measurement vector $\mathbf{M}$, the BA problem consists of minimizing the squared Mahalanobis distance:

$$\varepsilon^T \sum_M^{-1} \varepsilon$$

with $\varepsilon = \mathbf{M} - \mathbf{M}_0$ over $\mathbf{B}$. Applying the LM non-linear least squares algorithm to iteratively solve the problem, an equation with a regular sparse block structure is obtained, because of the lack of interaction between parameters of different cameras and different 3D points. Considerable computational benefits can be gained and a good solution can be finally obtained, especially when a good starting point for BA, i.e. obtained from the previous matrix estimations of $\mathbf{R}$ and $\mathbf{t}$, is given. In conclusion, Bundle Adjustment does not only increase the accuracy of the camera trajectory, but also prevents error build up in a way that decreases the frequency of total failure of the camera tracking.

3 Results

In order to test the overall performances of the previous described algorithms, a piece of C++ code has been developed and compiled either in standalone executable, either in Dynamic Link Library (DLL) form.

3.1 Real-time performances

A big effort has been dedicated to improve the performances of this algorithm and to enforce the robustness of the method, the following modifications were introduced:

- Threaded code was used to exploit multi-core CPUs;
- The calibration matrix $\mathbf{K}$ was obtained by the analysis of five different images, so a good compromise between precision and performances was achieved;
- The angle between two following images processed was kept fixed around 40 degrees so the Optical Flow performed a fast evaluation of the corresponding points. This choice was related to the fact that in the case of a video sequence, consecutive frames are very close together and the computation of epipolar geometry could be ill conditioned due to the short baseline;
- The conditioning number of the problem of finding matrix $\mathbf{F}$ can be augmented by an appropriate rescaling of the 2-D points extracted from the images;
- The RANSAC algorithm [4] was used to compute the fundamental matrix $\mathbf{F}$; RANSAC method is capable of smoothing data containing a significant percentage of gross errors, and thus is ideally suited for applications in automated image analysis, where interpretation is based on the data provided by error-prone feature detectors. Thanks to this method, the outliers contained in the initial set of feature points were detected and isolated, avoiding the least-squares approach to fail.
- For the Optical Flow computation, the Lucas-Kanade algorithm was chosen because of its robustness in presence of noise.

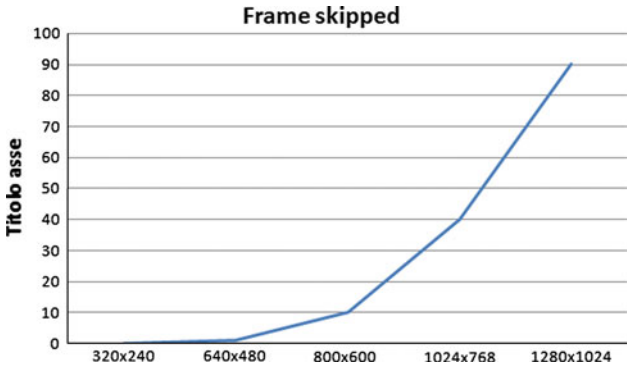

Fig. 7 Frames skipped during the session

- In the Bundle Adjustment process the Jacobian of the projection function was explicitly specified in order to reduce the computational cost of the method;
- The Bundle Adjustment was applied to the projection matrices and not only to the 3D points, to improve the result accuracy;
- Final 3D point were expressed with respect to the first camera reference system, so the number of changes of the coordinate system was reduced, in order to limit possible propagation of numerical errors.

3.2 Experimental

The proposed camera registration and 3D reconstruction method has been extensively validated with the aid of both synthetic and real image sequences. A low cost system composed by an elementary video device (such as a camera or a webcam) was used to acquire a video sequence of a fixed scene while a multi-core tablet PC was used for image processing and data analysis.

The reconstruction performances are also greatly affected by the camera resolution that, on one hand improves the feature identification and recover, but on the other hand induces a non linear performance reduction. In Fig. 7 an experimental data graph has been introduced in order to appreciate the relation between the camera picture resolution and the number of frames skipped during the session.

In fact frames are automatically skipped by the software on the basis of a setup file defined by the user. The operator defines the correct frame rate from several experimental tests, keeping also in count the CPU power. An automatic procedure is under development.

4 Conclusions

The paper presents the design and implementation of real-time 3D camera registration and geometry reconstruction, realized in order to be perform with a singular handheld camera freely moved and combined within a CAD environment. SRE plug-in has been designed for performance optimization, consolidating and modifying several algorithms used in different 3D computer vision fields. The very relevant amount of calculations needed to perform feature extraction from camera pose and 3D points reconstruction over a real object has been solved as needed in most industrial applications, like robotics and digital visualization, beneath the precision is never better than 0.5–1 mm in 3D coordinates. On the other hand, SRE seems to be particularly useful in CAD reverse modelling, where interactivity, much more than precision, helps to overcome the well-known problem of the slow geometry reconstruction.

References

1. Bolles, R.C., Baker, H.H.: Epipolar-plane image analysis: a technique for analyzing motion sequences. In: IEEE 3rdWorkshop on Computer Vision: Representation and Control, pp 168–178 (1985)
2. Broida, T.J., Chandrashekhar, S., Chellappa, R.: Recursive 3D motion estimation from a monocular image sequence. IEEE Trans. Aerosp. Electron. Syst. **26**(4), 639–656 (1990)
3. Faugeras, O.D., Luong, Q., Maybank, S.: Camera self-calibration: theory and experiment. European Conference on Computer Vision, (1992)
4. Fischler, M.A., Bolles, R.C.: Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Commun. ACM **24**(6), 381–395 (1981)
5. Hartley, R.I.: An investigation of the essential matrix. GE internal report, (1995)
6. Horn, B.K.P, Schunck, B.G.: Determining optical flow. Artif. Intell. **17**, 185–203 (1981)
7. Kato, H., Billinghurst, M.: Marker tracking and HMD calibration for a video-based augmented reality conferencing system. In: Proceedings of the 2nd IEEE and ACM International Workshop on Augmented Reality (1999)
8. Kutulakos, K.J., Vallino, J.: Affine objects representation for calibration-free augmented reality. Virtual Reality Annual International Symposium (VRAIS '96), pp 430–436. (1996)
9. Lu, Y., Zhang, J.Z., Wu, Q.M.J., Li, Z.N.: A survey of motion-parallax-based 3-D reconstruction algorithms. IEEE Trans. Syst. Man Cybern. C Appl. Rev. vol. 34, no. 4, pp. 532–548 (2004)
10. Shariat, H., Price, K.E.: Motion estimation with more than two frames. IEEE Trans. Pattern Anal. Mach. Intell. **12**(5), 417–434 (1990)
11. Triggs, B., McLauchlan, P., Hartley, R., Fitzgibbon A.: Bundle adjustment—a modern synthesis. Vision algorithms: theory and practice, (2000)
12. Zhang, Z.: A flexible new technique for camera calibration. IEEE Trans. Pattern Anal. Mach. Intell. **22**(11), 1330–1334 (2000)