Using Vanishing Points for Camera Calibration and Coarse 3D Reconstruction from a Single Image

E. Guillou, D. Meneveaux, E. Maisel, K. Bouatouch
IRISA, Campus de Beaulieu
35042 Rennes cedex, France
e-mail: {eguillou | dmenevea | maisel | kadi } @ irisa.fr

Abstract

In this paper we show how to calibrate a camera and to recover the geometry and the photometry (textures) of objects from a single image. The aim of this work is to make possible walkthrough and augmented reality in a 3D model reconstructed from a single image. The calibration step does not need any calibration target and makes only four assumptions: The single image contains at least two vanishing points, the length (in the 3D space) of one line segment (for determining the translation vector) in the image is known, the principle point is the center of the image and the aspect ratio is fixed by the user. Each vanishing point is determined from a set of parallel lines. These vanishing points help determining a 3D world coordinate system R_o. After having computed the focal length, the rotation matrix and the translation vector are evaluated in turn for describing the rigid motion between R_o and the camera coordinate system R_c. Next, the reconstruction step consists in placing, rotating, scaling and translating a rectangular 3D box that must fit at best with the potential objects within the scene as seen through the single image. With each face of a rectangular box is assigned a texture which may contain holes due to invisible parts of certain objects. We show how the textures are extracted and how these holes are located and filled. Our method has been applied to different real (pictures scanned from books, photographs) and synthetic images.

Keywords

Calibration - Reconstruction - Rendering - Texture extraction

1 Introduction and motivation

Modeling the geometry and the photometry (Reflectance, BRDF, Transmittance, Light source intensity) of a real scene, for a purpose of walkthrough or augmented reality, is known as a very hard task. Indeed, a real scene contains objects of highly detailed geometry and of complex physical properties. In addition, the light-material interactions (scattering in a participating medium, glossy reflection, transmission through a non smooth glass, spectral nature of light, diffraction, etc.) still are difficult to simulate.

One solution to this problem is to make use of Image Based Rendering methods (IBR) [1, 2, 3, 4, 5, 6]. These methods needs the acquisition of a large number of images (for example for building mosaics). This is why Image Based Modeling techniques (IBM) [7, 8, 9, 10] are preferred to IBR ones. These techniques recover, from a set of real images, the geometry of either polygonal objects or geometric primitives with known models. To improve realism, textures are extracted and may be either view-dependent or not.

For most IBM techniques the camera calibration is needed. This operation is often performed by inserting a calibration target in the real scene. An impressive literature has addressed this problem as will be seen later on.

In this paper we show how camera calibration and 3D reconstruction can be achieved from a single image unlike the existing IBR and IBM techniques. The assumptions made by our method are: The single real image contains 2 vanishing points, one line segment (for determing the translation vector) in the image is known, the principle point is the center of the image and the aspect ratio is fixed by the user. No calibration target is needed, which makes the method more general. The calibration of the camera consists in computing the focal length, the rotation and translation matrices with respect to a coordinate system defined by the vanishing points. Once the calibration has been performed the reconstruction algorithm is invoked. This algorithm tries to bound objects in the image with 3D rectangular boxes that can be rotated, scaled and translated. Each visible face of a box is assigned an extracted texture representing the appearance of the objects through its associated box. Note that the reconstruction in question is coarse but it is sufficient for our targeted applications. Next, VRML and PovRay descriptions of the reconstructed scene are generated which makes possible to walk through it or to generate images from novel viewpoints. Note that hitherto the box faces, not visible in the input image, have not been assigned any texture. We fill these holes with image patterns picked up from the image.

The rest of this paper is organized as follows. Section 2 outlines the contents of the existing papers related to our work. The calibration process is detailed in section 3 while the 3D reconstruction step is depicted in section 4. Before concluding, implementation details and results are given in section 5.

2 Prior works

For various computer vision applications, it is important to calibrate cameras [11]. Calibrating a camera consists in determining its intrinsic parameters (focal length, pixel ratio, center of projection) and its extrinsic parameters (rotation and translation expressing the position and orientation of the camera with respect to the world coordinate system). There are many existing techniques for solving this problem, among which very efficient calibration techniques have been proposed by Tsai [12] and Faugeras and Toscani [13]. Other techniques exploiting the presence of vanishing points in the image have been devised. These techniques will be called, from now on, *Vanishing Point Based Calibration* techniques or VPBC. In this section we briefly summarize only the VPBC calibration techniques since they are related to our work. We also outline the contents of four papers [14, 15, 16, 17, 18] dealing with 3D reconstruction from a single picture or a single panoramic image mosaic.

2.1 VPBC methods for camera calibration

Haralick's method [19] does not make explicitly use of vanishing points but of a similar idea that is the presence of a rectangle in the scene whose perspective projection is visible on the image plane. His method makes the following assumptions: The focal length is known, the center of projection is located at the origin of the world coordinate system (WCS), the camera performs only rotations. The rotation of the camera is decomposed into 3 rotations about each axis of the WCS.

In their paper [20], Caprile and Torre calibrate a camera from two or more images. In a first step, the focal length and the location of the intersection between the optical axis and the image plane are recovered from a single image of a cube (calibration target). In a second step, the rotation matrix and the translation vector which describe the rigid motion between the coordinate systems fixed in the two cameras are estimated from an image stereo pair of a suitable planar pattern (calibrating pattern). By matching the corresponding vanishing points (3 in this case) in the two images the rotation matrix is computed and the translation vector is estimated by means of a simple triangulation.

Shigang et al. [21] proposed a calibration method applied to a robot with a vision sensor moving around an indoor environment. The following assumptions are made: The robot moves on a flat horizontal plane and so does the camera, the axis of which is kept horizontal and perpendicular to the camera velocity, the environment contains many straight lines, the camera velocity and the focal length are known. The method first extracts horizontal lines from a sequence of images and then estimates the camera rotation from the motion of the corresponding vanishing points.

Chen and Jiang[22] make only one assumption: The focal length is known. They put a calibrating pattern (which is a uniform grid) in the environment and determine the lines defined by this pattern. Once these line have been extracted, they classify them into two groups: Horizontal and vertical lines. From the vanishing points associated with these groups of lines and assuming two line directions have been marked on the grid pattern (corresponding to the x and y axes of the WCS), the rotation matrix is determined. Then the translation vector is computed using the location of the lines of the grid pattern.

In Wang's and Tsai's work [23], a hexahedron is employed as a calibration target to generate a vanishing line of the ground plane from its projected image. The vanishing line includes useful geometric hints about the camera orientation parameters and the focal length, from which the orientation parameters can be computed analytically. As for the camera position parameters, they can be recovered using geometric projective relationships.

In a more recent paper Liebowitz et al. [17] make use of 3 vanishing points determined from three sets of parallel lines available in architectural scenes. The camera is calibrated from the rectification parameters of a set of planes. Their method does not need to estimate the rotation and translation parameters of the camera.

2.2 3D Reconstruction from a single image or a single panoramic image mosaic

To our knowledge four papers address the problem of 3D reconstruction from a single image [14, 16, 17, 18] and one from a single panoramic image mosaic [15].

The method, described in [14] does not employ any conventional computer vision technique but makes use of a spidery mesh to obtain a very simple scene model from a single image using a user interface. The aim is to provide a new type of visual effects for various animations rather than recovering a 3D model for walkthrough. The method starts by specifying one vanishing point in the image. Then the geometry of the background is approximated with a model composed of 5 rectangles. This model is a polyhedron-like form with the vanishing point on its base. Each foreground object (near the viewer) is modeled with hierarchical polygons. The billboard-like representation is used for the foreground objects.

Parodi and Piccioli [16] have investigated the reconstruction of 3D models in a scene from line drawing (on a single image) using geometrical constraints provided by the location of vanishing points and an incidence structure I represented by a graph whose nodes are vertices and planar panels. A spatial structure $S = (I, T)$ is also determined, where T is a set of depth relations. The connected components of this graph are determined. A component can be seen as a different object or a set of physically connected objects. Given the location of a 3D vertex, the authors propose a formula which determines the 3D coordinates of other vertices belonging to the same planar panel. This formula contains some terms depending on two vanishing points determined from the planar panel. The method computes the vanishing points associated with each planar panel. The focal length is calculated using the algorithm described in [20]. A scale factor is associated with each component. To check the realizability of the reconstruction, three independent tests are performed in sequence. In addition, one of these tests checks if the set of scale factors can be found such that the resulting 3D locations of all vertices simultaneously satisfy the incidence and depth relations.

In [15], Shum et al. presents an interactive modeling system that constructs a 3D model from one or several panoramic image mosaics. The system recovers the camera pose for each mosaic from known line directions and points, and then constructs the 3D model using all available geometrical constraints (soft or hard). The modeling process is formulated as a linearly-constrained least squares problem which can be solved with QR factorization method. The method makes use of parallel lines and/or known points to recover the camera rotation. For a single panorama, the translation is set to zero if no point in the 3D model is given. Textures are also extracted following the method in [24].

Liebowitz et al. [17] propose a two-pass reconstruction method applied to planar facades. In a first step the visible facades are individually reconstructed by rectification. The second step proceeds in an incremental fashion, that is to say, the facades are positioned in turn. More precisely, a facade A can be positioned relative to another facade B only if A and B share common points. The orientation of the facade B as well as its size are determined using the common points and the rectification parameters of the planes supporting the two facades A and B.

Van Den Heuvel [18] describes a method for reconstructing, from a single image, polyhedral object models using measurements (image lines, object information). Topology (of coplanarity type) and geometry (parallelism, perpendicularity, symmetry) constraints are used by the reconstruction process. The method assumes the camera to be calibrated.

2.3 Discussion

The VPBC methods described above have the advantages of requiring a few images and a low computing time. However either they make a certain number of assumptions or require placing in the scene a calibration target or a calibrating pattern. These methods are unusable in the case the user is provided with images to be calibrated but does not have the possibility to access the real 3D scene for including a calibrating pattern. To remedy this, as in [15, 16, 17], the calibration method described in this paper needs as input data only one image (a picture) containing two vanishing points instead of three as required by most of the existing VPBC techniques. It does not need any calibrating pattern nor target.

Recall that [14] provides a means for extracting an approximated 3D model of the scene background and representing the foreground objects with billboards from a single image to make a tour into the picture without the need of camera calibration nor actual 3D reconstruction. With this technique the set of allowed trajectories is limited.

In [16], vanishing points are computed for each planar panels, incidence and spatial structures are determined and the realizability of the recovered models are checked using three independent tests. We think that these processings require a high computing time limiting interactivity (this paper does not give any result on the computing time). Unlike this method, our approach needs only two vanishing points for all the scene and does not require any realizability check since our 3D model is coarse (a rectangular box). Moreover, we do not need to determine incidence and spatial structures but only to draw two sets of parallel lines on the single image.

In [15, 16], to avoid any ambiguity on the determination of the rotation matrix, the authors suggest to select points of known coordinates. For recovering a model with 6 faces, the system takes 15 minutes. For recovering a model of moderate complexity, we think that the modeling time would be far higher. Unlike this method, our approach allows to recover, without ambiguity, the rotation matrix using only two sets of parallel lines, each set defining a vanishing point. Besides, our approach operates on a single image not a panoramic image mosaic. On the other hand, their reconstruction method is more precise than ours.

In [18], the reconstruction process relies on the extraction of straight lines from which line segments (corresponding to edges of objects) are selected. Topological relations between the line segments have to be specified by the user. These relations allow to close polylines corresponding to the border of the face of an object. A set of constraint equations are determined and solved to help reconstructing the objects as seen through the single image. This method seems promising but it assumes that the camera is calibrated (intrinsic parameters).

Although they have been done independently, the most recent work described in [17] and our work have several similarities: extraction of vanishing points from sets of parallel lines on the single image, no calibration target, reconstruction from a single image. However some differences exist. The first one concerns camera calibration. Indeed, unlike [17], our method determines the camera pose, say the translation vector and the rotation matrix. This allows to reconstruct each primitive object independently of the others. Furthermore, our reconstruction method is not limited to facades.

We want to propose a method for calibrating precisely the used camera and recovering the geometry (coarsely) and the photometry (by means of extracted textures). This method must allow an easy walthrough (with a large set of possible trajectories) in the reconstructed 3D model rather than in the picture. Unlike in [17], where some skills about computer vision and projective geometry are required as background, our main goal is to propose a simple and fast reconstruction system that can be within the capability of any researcher in computer graphics having no skill in the field of computer vision.

3 Calibration using two vanishing points

In what follows we make the following assumptions: The image contains two vanishing points, the principal point is the center of the single image, the aspect ratio is fixed by the user. We will see that the translation vector will be determined up to a scale factor if we do not know the length of one line segment in the 3D space. In this section we deliberately give details about several points of our calibration method to make this paper clear and understandable for a computer graphics reader having no skill in computer vision. These details can be skipped by a computer vision reader.

3.1 Vanishing point, line and plane

Let D be a line in the 3D space passing through a point $Q_0 = (a_x, a_y, a_z)^t$ and of direction $d = (d_x, d_y, d_z)^t$. Its parametric equation is: $Q = Q_0 + l \cdot d$, where l is a parameter and $Q = (x, y, z)^t$.

The vanishing point associated with D is the point $F_D = (x_\infty, y_\infty, z_\infty)^t$ on the image plane. Its coordinates are:

$$\begin{cases} x_\infty &= \lim_{l\to\infty} f\frac{a_x+l.d_x}{a_z+l.d_z} &= f\frac{d_x}{d_z} \\ y_\infty &= \lim_{l\to\infty} f\frac{a_y+l.d_y}{a_z+l.d_z} &= f\frac{d_y}{d_z} \\ z_\infty &= f \end{cases} \tag{1}$$

All the lines parallel to D define the same vanishing point F_D. The vanishing line L_v associated with a plane Π contains the vanishing points defined by all the lines supported by this plane. From now on, the plane passing through the center of projection of the camera and containing L_v is called vanishing plane associated with Π.

3.2 Determining the focal length from a single image

We consider a pinhole camera model. The center of projection is O and its projection onto the image plane is P. Suppose the image contains two sets of parallel lines defining two vanishing points. Suppose the lines of the first set are perpendicular to those of the second. Take two lines in each set so that their intersection is a rectangle $ABCD$ in the 3D space. Let us call $abcd$ its perspective projection (see figure 1). Let $\overrightarrow{u}$ and $\overrightarrow{v}$

be the direction vectors of the lines (AD) and (AB) respectively. Recall that $(AD) \parallel (BC)$ and $(AB) \parallel (DC)$. We suppose that the rectangle $ABCD$ is not parallel to the image plane. The lines (AD) and (BC) define the same vanishing point $F_u = (ad) \bigcap (bc)$ on the image plane. In the same way (AB) and (DC) have a vanishing point $F_v = (ab) \bigcap (dc)$. We suppose the coordinates of F_u and F_v, expressed in the camera coordinate system, are known.

Let $\Pi(O, \overrightarrow{OF_u}, \overrightarrow{OF_v})$ be the vanishing plane associated with $ABCD$. The vanishing line defined by $ABCD$ is the line (F_uF_v) which is also the intersection between $\Pi(O, \overrightarrow{OF_u}, \overrightarrow{OF_v})$ and the image plane. Let P_{uv} the orthogonal projection of P on the the line (F_uF_v). We want to compute the focal length:

$$f = OP = \sqrt{{OP_{uv}}^2 - {PP_{uv}}^2}.$$

Since the coordinates (in the camera coordinate system) of P, F_u and F_v and the length of PP_{uv} are known, only the length of OP_{uv} has to be calculated for evaluating the focal length.

We suppose the triangles OF_uF_v, OF_uP_{uv} and OF_vP_{uv} are right-angled with $\hat{O}$, $\hat{P_{uv}}$ $\hat{P_{uv}}$ as the right angles respectively (see figure 2). We get then:

$$\left\{ \begin{array}{rcl} \alpha + \gamma & = & \frac{\pi}{2} \\ \beta + \theta & = & \frac{\pi}{2} \\ \gamma + \theta & = & \frac{\pi}{2} \end{array} \right. \tag{2}$$

From equation 2 we derive:

$$\left\{ \begin{array}{rcl} \alpha & = & \theta \\ \gamma & = & \beta \end{array} \right. \tag{3}$$

Consequently, as $\alpha = \theta$ and the triangles OF_uP_{uv} et OF_vP_{uv} are right-angled with $\hat{P_{uv}}$ as the right angle, then:

$$\frac{OP_{uv}}{P_{uv}F_u} = \frac{F_vP_{uv}}{OP_{uv}} \tag{4}$$

We get then:

$$OP_{uv} = \sqrt{F_vP_{uv} * P_{uv}F_u} \tag{5}$$

Let us prove now that the triangles OF_uF_v, OF_uP_{uv} and OF_vP_{uv} are right-angled. Since F_u (resp. F_v) is the vanishing point associated with the direction $\overrightarrow{u}$ (resp. $\overrightarrow{v}$), then $\overrightarrow{OF_u} = k_u \overrightarrow{u}$ (resp. $\overrightarrow{OF_v} = k_v \overrightarrow{v}$). By définition, $\overrightarrow{u} \cdot \overrightarrow{v} = 0$, thus $\overrightarrow{OF_u} \cdot \overrightarrow{OF_v} = 0$. Consequently $(OF_u) \perp (OF_v)$, which proves that the triangle OF_uF_v is right-angled with $\hat{O}$ as the right angle.

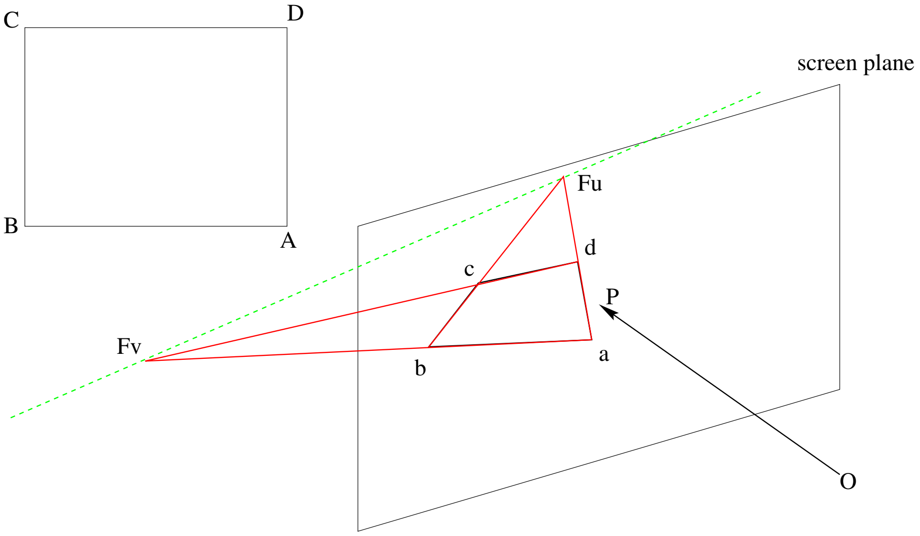

Figure 1: Focal length

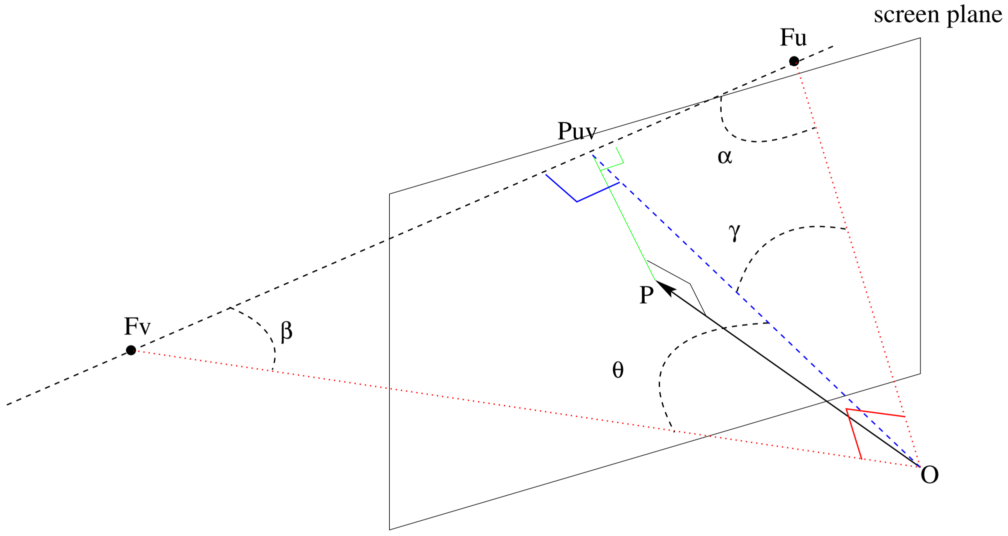

Figure 2: Focal length

By construction, $\overrightarrow{OP} \cdot \overrightarrow{F_uF_v} = 0$ and $\overrightarrow{PP_{uv}} \cdot \overrightarrow{F_uF_v} = 0$. We can write $\overrightarrow{(OP_{uv})} \cdot \overrightarrow{F_uF_v} = \overrightarrow{OP} \cdot \overrightarrow{F_uF_v} + \overrightarrow{PP_{uv}} \cdot \overrightarrow{F_uF_v} = 0$, hence the lines (OP_{uv}) and (F_uF_v) are perpendicular. The triangles OF_uP_{uv} and OF_vP_{uv} are then right-angled with $\hat{P_{uv}}$ as the right angle.

3.3 Computing the rotation matrix

The problem now is to compute the rotation matrix which describes the rigid motion between the world and camera coordinate systems which are called R_o and R_c respectively. R_o may have as an origin any point in the 3D scene (the point A for example). It is defined by 3 basis vectors $\overrightarrow{u}$, $\overrightarrow{v}$ (see subsection 3.2) and $\overrightarrow{w}$ (such that $\overrightarrow{w} = \overrightarrow{u} \times \overrightarrow{v}$). See figure 3 for further details. We can write $R_o = (A, \overrightarrow{u}, \overrightarrow{v}, \overrightarrow{w})$ and $R_c = (O, \overrightarrow{i}, \overrightarrow{j}, \overrightarrow{k})$. The 2D coordinate system associated with the screen plane can be denoted $R_s = (P, \overrightarrow{i}, \overrightarrow{j})$, P being the orthogonal projection of the center of projection of the camera onto the screen plane.

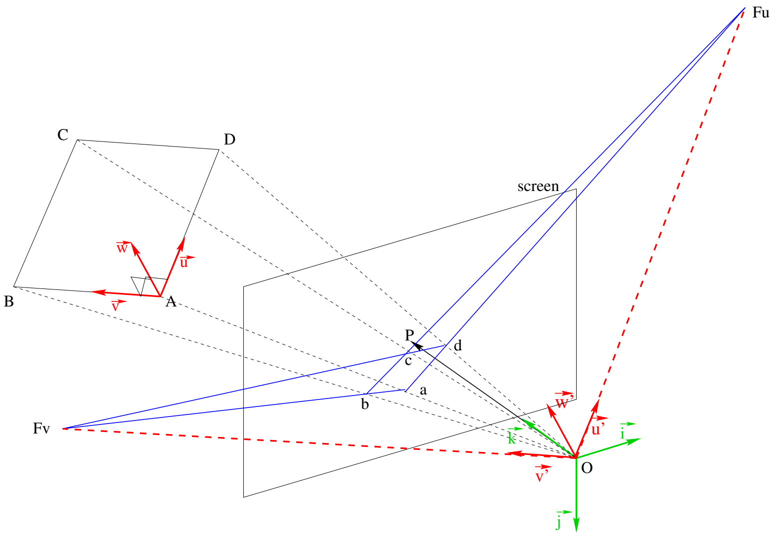

Figure 3: Determing the rotation matrix

We adopt the following notation $\overrightarrow{V}_{/R}$ which means that the vector $\overrightarrow{V}$ is expressed in the coordinate system R. The objective is to determine the rotation matrix $M_{o\to c}$ such that:

$$M_{o\to c} \bullet \overrightarrow{u}_{/R_o} = \overrightarrow{u}_{/R_c}\,, M_{o\to c} \bullet \overrightarrow{v}_{/R_o} = \overrightarrow{v}_{/R_c} \text{ and } M_{o\to c} \bullet \overrightarrow{w}_{/R_o} = \overrightarrow{w}_{/R_c}. \tag{6}$$

Let $D_{u'}$ and $D_{v'}$ be the lines passing through the center of projection O and aligned with the vectors $\vec{u'}$ and $\vec{v'}$ respectively (figure 3). Their respective vanishing points are then F_u and F_v. This proves that the basis vectors $(\overrightarrow{u}, \overrightarrow{v}, \overrightarrow{w})$ and $(\vec{u'}, \vec{v'}, \vec{w'})$ are similar. Consequently the same rotation describes the rigid motion between $R_c(O, \overrightarrow{i}, \overrightarrow{j}, \overrightarrow{k})$ and $R_o(A, \overrightarrow{u}, \overrightarrow{v}, \overrightarrow{w})$, and between $R_c(O, \overrightarrow{i}, \overrightarrow{j}, \overrightarrow{k})$ and $R_{o'}(O, \vec{u'}, \vec{v'}, \vec{w'})$. The coordinates of the vectors $\vec{u'}$, $\vec{v'}$ and $\vec{w'}$ in R_c are known:

$$\vec{u'}_{/R_c} = \frac{\overrightarrow{OF_u}_{/R_c}}{\|\overrightarrow{OF_u}_{/R_c}\|} = (\frac{F_{u_i}}{s_1}, \frac{F_{u_j}}{s_1}, \frac{f}{s_1})^t ; \vec{v'}_{/R_c} = \frac{\overrightarrow{OF_v}_{/R_c}}{\|\overrightarrow{OF_v}_{/R_c}\|} = (\frac{F_{v_i}}{s2}, \frac{F_{v_j}}{s2}, \frac{f}{s2})^t ; \vec{w'}_{/R_c} = \vec{u'}_{/R_c} \times \vec{v'}_{/R_c}, \tag{7}$$

with $\vec{w'}_{/R_c} = (w'_i, w'_j, w'_k)$, $s1 = \|\overrightarrow{OF_u}_{/R_c}\|$, $s2 = \|\overrightarrow{OF_v}_{/R_c}\|$ and f the focal length.

We can write:

$$M_{o\rightarrow c} \bullet \overrightarrow{u}_{/R_o} = \vec{u'}_{/R_c}, M_{o\rightarrow c} \bullet \overrightarrow{v}_{/R_o} = \vec{v'}_{/R_c} \text{ and } M_{o\rightarrow c} \bullet \overrightarrow{w}_{/R_o} = \vec{w'}_{/R_c}. \tag{8}$$

Knowing that $\overrightarrow{u}_{/R_o} = (1,0,0), \overrightarrow{v}_{/R_o} = (0,1,0), \overrightarrow{w}_{/R_o} = (0,0,1)$ and using equations 7 and 8 we get:

$$M_{o\rightarrow c} = \begin{pmatrix} \frac{F_{u_i}}{\sqrt{{F_{u_i}}^2+{F_{u_j}}^2+f^2}} & \frac{F_{v_i}}{\sqrt{{F_{v_i}}^2+{F_{v_j}}^2+f^2}} & w'_i \\ \frac{F_{u_j}}{\sqrt{{F_{u_i}}^2+{F_{u_j}}^2+f^2}} & \frac{F_{v_j}}{\sqrt{{F_{v_i}}^2+{F_{v_j}}^2+f^2}} & w'_j \\ \frac{f}{\sqrt{{F_{u_i}}^2+{F_{u_j}}^2+f^2}} & \frac{f}{\sqrt{{F_{v_i}}^2+{F_{v_j}}^2+f^2}} & w'_k \end{pmatrix} \tag{9}$$

3.4 Computing the translation vector

Consider figure 4. Let A' be the perspective projection of A and $\overrightarrow{A'P'}$ that of the vector $\overrightarrow{AP}$ which is a vector parallel to the axis $\overrightarrow{u}$ of R_o and originating at A, the origin of R_o. To determine the translation vector we suppose that the length l of $\overrightarrow{AP}$ is known, otherwise the translation vector will be computed up to a scale factor.

We can write:

$$\overrightarrow{AP}_{/R_o} = l \cdot \overrightarrow{u} ; \overrightarrow{AP}_{/R_c} = M_{o\rightarrow c}.\overrightarrow{AP}_{/R_o}.$$

Let P'' be the intersection of the line (OP) with the line D passing through A' and whose direction vector is $\overrightarrow{AP}$. Then:

$$P''_{/R_c} = (OP)_{/R_c} \cap D = (OP')_{/R_c} \cap D \tag{10}$$

As the triangles $OA'P''$ and OAP are similar then from Thales' theorem we get:

$$\frac{\|\overrightarrow{A'P''}_{/R_c}\|}{\|\overrightarrow{AP}_{/R_c}\|} = \frac{\|\overrightarrow{OA'}_{/R_c}\|}{\|\overrightarrow{OA}_{/R_c}\|} \tag{11}$$

or:

$$\|\overrightarrow{OA}_{/R_c}\| = \frac{\|\overrightarrow{OA'}_{/R_c}\|.\|\overrightarrow{AP}_{/R_c}\|}{\|\overrightarrow{A'P''}_{/R_c}\|} \tag{12}$$

hence:

$$\overrightarrow{OA}_{/R_c} = \|\overrightarrow{OA}_{/R_c}\|.\frac{\overrightarrow{OA'}_{/R_c}}{\|\overrightarrow{OA'}_{/R_c}\|} \tag{13}$$

The translation vector is:

$$\overrightarrow{T_{o\rightarrow c}}_{/R_o} = M_{c\rightarrow o}.\overrightarrow{OA}_{/R_c} \tag{14}$$

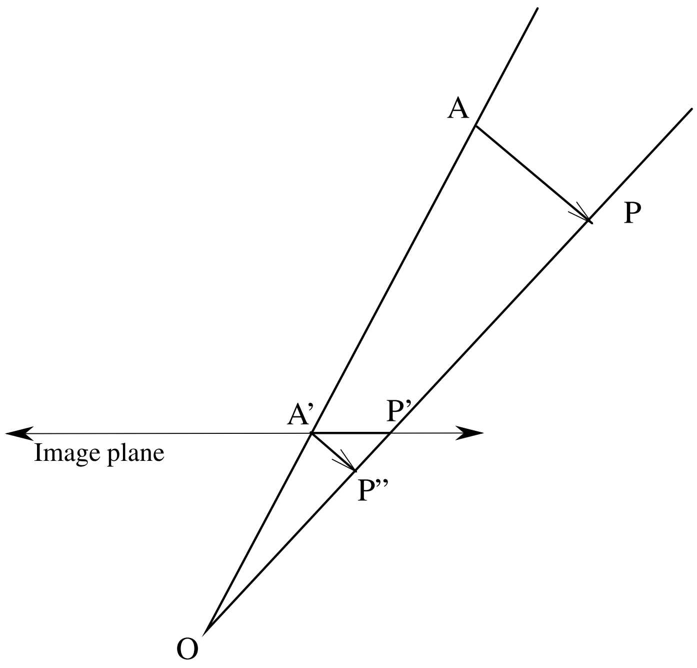

Figure 4: Determining the translation vector

3.5 Discussion

As the vanishing points are determined manually, one has to compute them with precision. For this reason the resolution of the used single image must be high. Note that vanishing points can be extracted automatically [25, 26] and our method is not limited to pose estimation since it determines the focal length. Moreover, as we are interested in coarse 3D reconstruction for walkthrough, we can neglect camera distortions and assume that the principle point is close to the image center.

4 Coarse 3D reconstruction

4.1 Principle

Once the calibration step has been completed, the user can start the reconstruction process from the single image. It operates as follows.

First we generate a planar reference grid (with a fine resolution) supported by the uv plane of the world coordinate system R_o. As the camera has been calibrated and R_o determined, a unit cube is placed in the scene and displayed on the image (by composition) . The lower left vertex of its front facing face is placed at the origin of R_o. This cube is moved, scaled and rotated with respect to the u, v and w axes so as to bound an object in the scene. The result is a parallelepiped which fits as best as possible with the object. These geometric transformations are facilitated thanks to the grid. See figure 9 for further details. This operation results in a set of rectangular boxes, each one bounding an object as seen in figure 10. Each face of a cube is assigned a texture corresponding to the image of the associated object as seen by the viewer (center of projection). Note that a cube's face may partially be occluded by the faces of other objects, which generates holes in the textures. These holes are filled with other textures extracted from the image.

4.2 Placing the unit cube

For each potential object, the planar reference grid is placed on the uv plane of the world coordinate system R_o thanks to a dialog box (generated by our user graphics interface) which allows the user to enter five parameters: the grid resolution and its size given by $u_{min}, u_{max}, v_{min}, v_{max}$. Then this grid is placed just under the object to be reconstructed by moving it along the w axis using two different keys. The coordinates of the grid points are displayed onto the image. By using another dialog box the user can translate, rotate and scale the unit cube (originally placed in such a way the lower left corner of its front face coincides with the origin of R_o) so as to fit with the object. Note that objects such as ceiling and floor are bounded with rectangular boxes of very small thickness.

4.3 Texture extraction

A face of a rectangular box is projected onto the image plane. The result is a quadrilateral to which is assigned a rectangular texture as explained in figure 5. Let x' and y' be the texture coordinates. $x' \in [0,1], y' \in [0,1]$. The 2D projective transformation $H = (h_{ij})$ mapping a point $P' = (x', y', 1)^t$ of the texture to a point $P = (x, y, 1)^t$ of the image is computed as follows (see figure 5 for the notations). Each vertex $P'_i = (x'_i, y'_i, 1)^t$ of the texture is transformed to a vertex $P_i = (x_i, y_i, 1)^t$ of the projected box's face as: $P_i = H \bullet P'_i$. More

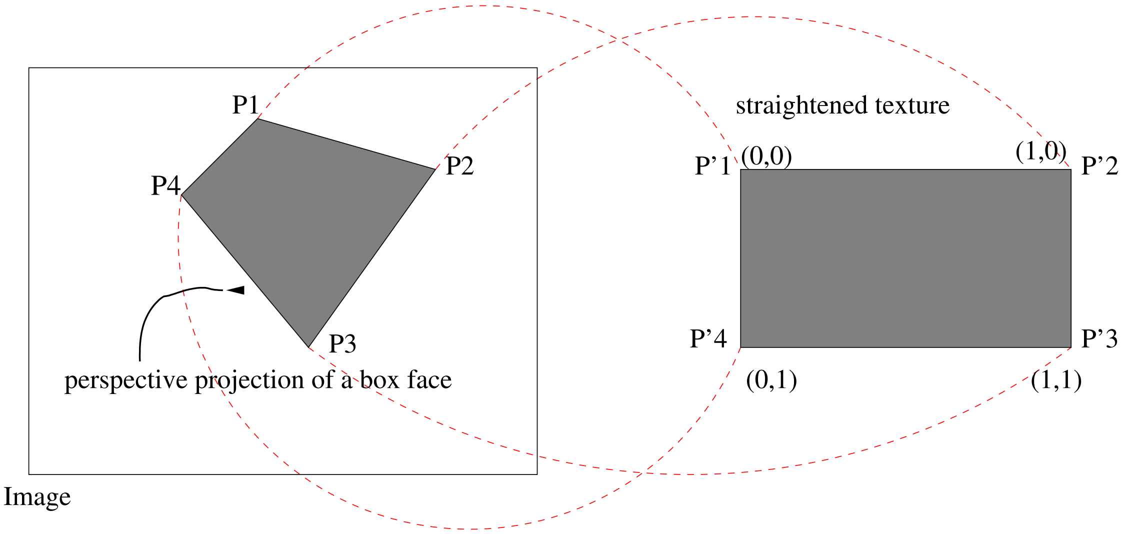

Figure 5: Texture extraction and rectification: the texture coordinates range from 0 to 1.

precisely [4]:

$$x_i = \frac{h_0 x'_i + h_1 y'_i + h_2}{h_6 x'_i + h_7 y'_i + 1},\; y_i = \frac{h_3 x'_i + h_4 y'_i + h_5}{h_6 x'_i + h_7 y'_i + 1} \tag{15}$$

with

$$H = \begin{pmatrix} h_0 & h_1 & h_2 \\ h_3 & h_4 & h_5 \\ h_6 & h_7 & 1 \end{pmatrix} \tag{16}$$

Note that this mapping operates on homogeneous coordinates, (x, y) and (x', y') being cartesian coordinates. We get then a system of equations with the h_js as unknowns. Next, each texel of the texture is mapped (using H) to a pixel of the image and is assigned the color of this pixel. The result is a rectified rectangular texture. To avoid several pixels to be mapped to the same pixel of the image, the resolution (in pixels) of this texture is chosen as the number of pixels covered by the smallest edge of the projected box face for each coordinate axis of the texture. For example, in the case of figure 5, these edges are P_1P_4 (since $P_1P_4 < P_2P_3$) and P_1P_2 ($P_1P_2 < P_4P_3$). The resolution is then the number of pixels covered by each of these edges.

Note that the extracted texture of each box's face depends on the single view. Thus it is view-dependent. When walking through the reconstructed 3D model, the texture mapping process uses only this single texture even though the camera changes position and orientation. This is due to the fact that only one image is available. This situation can raise the following problem. Let us take as an example the chandelier in figure 12. As the faces of the box associated with the chandelier are opaque, certain pieces of the background that should be visible from a novel viewpoint are occluded by these faces. To remedy this, the textures associated with the faces visible from the original viewpoint are modified as follows. The portions of the textures that do not correspond to the chandelier are made transparent (using our graphics user interface) with the help of the α channel. The obtained result is shown in figure 12.

In the case the user is provided with several photographs of the same scene taken from different viewpoints, the faces of all the boxes will be assigned several view-dependent textures, one for each viewpoint. To render the 3D model from a novel point of view, texture mapping would be performed using a weighted combination of the extracted textures as explained in [7]. Another solution is to extract view-independent textures from multiple photographs [9].

4.4 Detecting and filling the holes

The holes correspond to the portions of the box faces that are invisible from the viewpoint associated with the single image. The hole-filling problem has already been addressed in [7, 27, 28, 8]. Since all the proposed algorithms make use of multiple images they cannot be applied to our case (a single image).

This is why we have implemented our own method which is described hereafter. Consider figure 6 to explain how holes in the textures are determined and filled. Let A and B be two boxes. To see whether B occludes A or not, each box is projected onto the screen plane. The result is two polygons on the screen. Using a min-max technique we compute for each of these two polygons a rectangular bounding box aligned with the screen axes. If the two resulting bounding boxes intersect then a face of B could occlude a face of A. In this case, to see if a face F_A of A is occluded by a face F_B of B, we project these two faces and compute their intersection I on the screen plane. If this intersection is not empty, then we check if F_B occludes F_A by tracing a few rays originating at the viewpoint and passing through I. In the case of occlusion, I is an occlusion polygon corresponding to a hole in the texture assigned to the face F_A. This hole detection process is repeated for all the faces of B and all the other potential occluding boxes. For each box face, the results is a set of occlusion polygons lying on the screen. To fill these holes, our method proceeds as follows. Let F be a face containing holes. We project F onto the screen and scan-convert the associated occlusion polygons with a black color. Then we extract from the initial image one or several appropriate image patterns (using our graphics user interface) that we use to fill the black regions that correspond to the holes. Next, we construct the new texture assigned to F as explained above.

Let us consider the figure 9 and more precisely the chimney to show how image patterns are extracted for filling the holes. In this figure (the single image) we see only the front facing part and the right side of the chimney, while the left side is occluded. When placing a box around the chimney, the left side of the chimney corresponds to a large hole that we can fill with the visible texture associated with the right face of the chimney. We suppose in this case that the right and left faces of the chimney are similar.

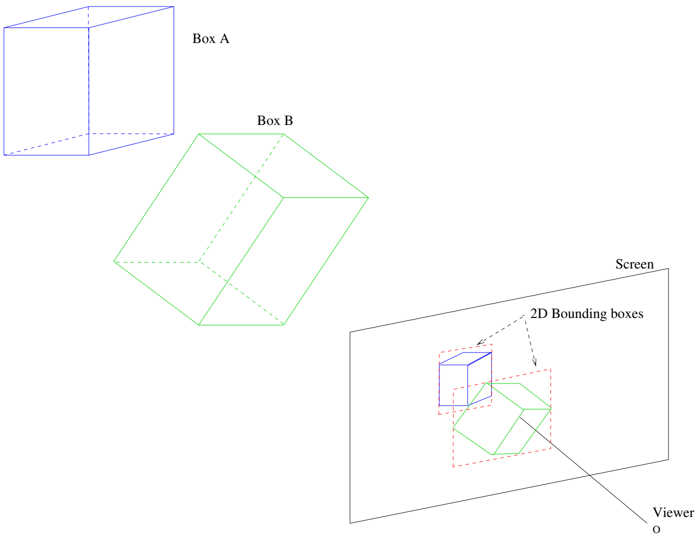

Figure 6: Object *B* occludes object *A*

4.5 Discussion

At present, our reconstruction method does not handle precisely cast shadows. This due to our technique of filling holes with image patterns extracted from the image. Trees and any vegetation can be coarsely reconstructed, which makes our method applicable to any kind of scene: indoor and outdoor.

5 Results

Our method has been applied to various real and computer generated images. Let us see now some results obtained for the image of figures 9, 10,11. This image has been scanned from a book. Its resolution is $960x612$. Calibration and Reconstruction took less than one hour and 30 boxes have been determined to bound the potential objects. Our system has been implemented with Matlab. We expect a lower reconstruction time with other programming languages such as C or C++. The calibration step is very fast. Placing boxes around the potential objects is a simple and fast operation. On the other hand, the texture extraction step is computationally expensive compared to the other steps (more than 60% of the total modeling time).

The figure 11 shows the two sets of lines (blue and green) giving the two vanishing points needed by the calibration and the reconstruction steps. For each vanishing point the user selects (with the mouse) a set of potential parallel lines. To make this selection easier the user can zoom in the portions of the image containing these lines. We prefer to select a maximum number of lines for a set, for more precision. Note that vanishing points can be extracted automatically using the method described in [25, 26]. Once the vanishing points have been determined we proceed by displaying a 2D grid on the image to guide the user for placing a 3D box around an object as seen in figure 9. In this figure we see a blue box bounding a chair. This process is repeated for all the potential objects seen in the image. The result is a set of boxes as shown in figure 10. In this figure the boxes have been displayed with different colors for a reason of clarity. Afterwards, textures are extracted and assigned to the faces of all the boxes. Next, the user selects (with the mouse) and extracts appropriate portions of the image that will be used for filling the holes in the faces' textures.

The extracted 3D model is converted into the VRML and PovRay formats, and finally saved into two files. These files will allow an easy walkthrough (using a VRML browser) and a generation of other images from different viewpoints (figure 12).

The image given in figure 13, acquired with a classical camera, has a resolution of $392x512$. It has required 31 boxes to bound the potential objects. Calibration and reconstruction took less than 2 hours. Figure 14 is an image of the reconstructed model as seen from a novel viewpoint.

Recall that our camera calibration method relies on vanishing points determined by manually selecting lines on the single image. This manual selection may affect the precision of the recovered camera parameters and the image coordinates. Measuring this precision amounts to compare the actual values of the parameters with the recovered ones. Unfortunately only the actual value of the focal length may be available. Let us now see how we can assess the precision of the recovered focal length in the case of image 15. The actual focal length of this image is $2.8cm$. We first determine manually a set of parallel lines (defining a vanishing point) with a maximum of precision by zooming the single image. We find a focal length equal to 2.8051cm. Let us now simulate errors in the process of selecting parallel lines. Each of these lines is represented by a ray (of same direction) originating at any point lying on this line (for instance the left most point). Next, for each ray its origin is randomly moved away from its initial position along the vertical image axis. Each displacement is expressed in terms of pixels. As a result, the lines associated with a same vanishing point undergo displacements of different values whose average is D_{ave}. Then, the vanishing point associated with these moved lines is determined. The same process is used for determining the second vanishing point. We compute then the associated focal length. This overall process is repeated for several D_{ave} as shown in figure 7 which gives the variation of the calculated focal length (blue curve) with the average value D_{ave} of the displacements of the ray origins. In this figure the green curve represents the variation of the error on the calculated focal length. We can see that a displacement around the right position of the ray origins of about -1.5 to 3 pixels does not affect the precision of the calculation of the focal length. This proves that our manual selection of parallel lines is sufficiently precise.

Another way of simulating errors in the process of computing vanishing points is to move one vanishing point away from its precise position (determined by zooming the single image 15 as explained above) in a window of $500x500$. After each displacement of this vanishing point we compute the focal length. The result is plotted in figure 8. In this figure the blue curve represents the calculated focal length as a function of the average value of the displacements of the vanishing points. We can see that an error of 50 pixels is hardly noticeable since it entails an error (green curve) of $0.05cm$ on the calculated focal length.

It is true that any error on the camera calibration (due to the manual selection of sets of parallel lines and to the assumptions made) would affect the 3D reconstruction of potential objects with rectangular boxes. In our case this problem is not crucial since the position of the rectangular boxes bounding objects can be interactively modified and adjusted to compensate any calibration error.

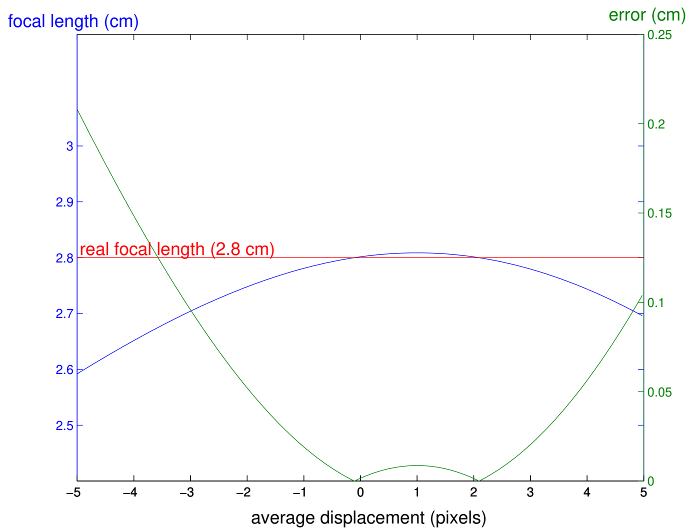

Figure 7: Error on focal length estimation: displacement of parallel lines.

6 Conclusion and future work

The work presented in this paper contains two contributions: camera calibration and 3D reconstruction. The first one allows the calibration of a camera from a single image (a photograph or any kind of image) containing two vanishing points. The used method determines the focal length, then the rotation and translation matrices describing the rigid motion between the world and camera coordinate systems. To compute

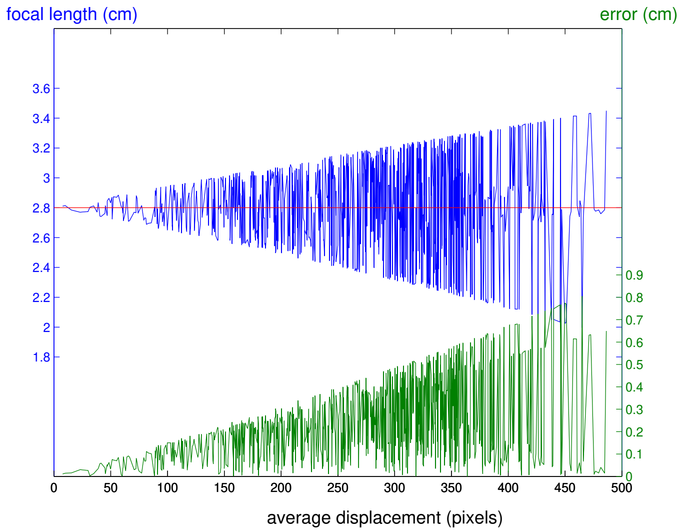

Figure 8: Error on focal length estimation: displacement of one vanishing point.

precisely the translation matrix, we need to know the length of a segment (or edge) in the real scene. To determine the two vanishing points, the user selects two sets of parallel lines. Each set defines a basis vector of the world coordinate system: $\vec{u}$ and $\vec{v}$. In the current implementation, the two sets of lines are chosen such that $\vec{u} \perp \vec{v}$. Thus, the third basis vector is computed as: $\vec{w} = \vec{u} \times \vec{v}$. Unlike the method described in [14], our approach makes use of two vanishing points (rather than one) and recovers 3D models in a more accurate way. Any kind of image can be used, not necessarily a panoramic image mosaic as in [15].

Once the calibration has been performed, reconstruction is effected by placing around each potential object a rectangular box, the faces of which are assigned textures. The holes in the extracted textures corresponding to invisible parts of box faces are filled with other appropriate portions of the image. These textures depend on the single viewpoint. The same textures are used when rendering the 3D reconstructed model from a novel viewpoint, which may be a source of artifacts as seen on the left side of figure 14.

Though the proposed calibration method seems robust, the reconstruction step has to be improved. One solution is to deform (using the graphics user interface) a rectangular box so as to fit at best with the object to be reconstructed, or to introduce other geometric primitives (such as: sphere, cylinder, polyhedron, etc) to bound the objects in the single image. With multiple images, reconstruction could be performed more precisely. Regarding texture mapping we can do better if multiple images are available as shown in [7, 9].

We are working on several extensions to enhance our system. For example, if several views are provided we can calibrate the camera from one image, select a certain number of points on this image and find the corresponding points on the other images. Next, we determine the fundamental matrices between pairs of images. From these matrices we compute the 3D transformations (rotation and translation) between pairs of views corresponding to the images. With these transformations, when placing, on one image, a box around a potential object, this box can be viewed on the other images. This has several advantages. First, box placement gets more precise. Second, a box's face is assigned several view-dependent textures. Third, we can enhance each texture by adding an out-of-plane displacement component at each texel to get sprites with depths as explained in [29]. Another improvement concerns our graphic user interface. This improvement must help the user to speed up the placement of boxes as well as texture extraction which requires, for each box's face, to outline (delineation) parts of the textures corresponding to the object as seen through its box and to make transparent (using the alpha channel) the other parts.

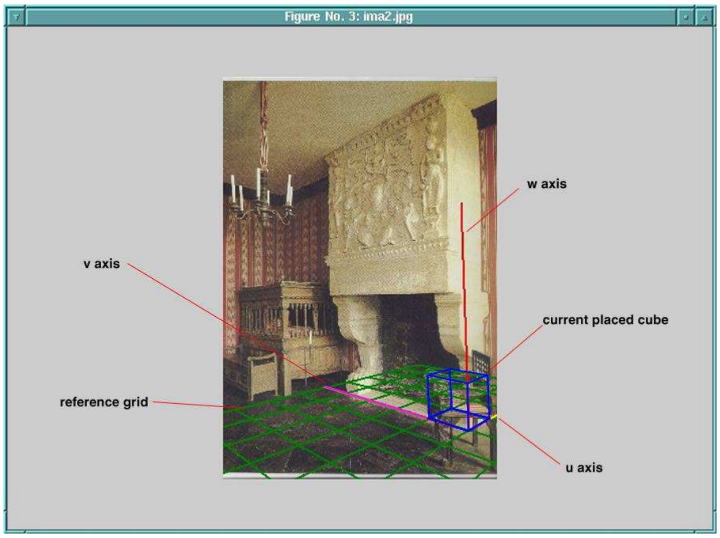

Figure 9: The grid (green) and a box bounding a chair (blue)

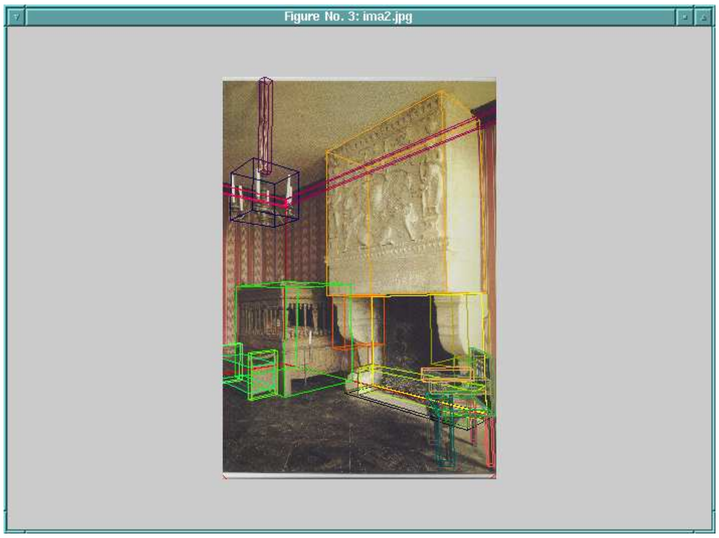

Figure 10: All the extracted 3D boxes with different colors

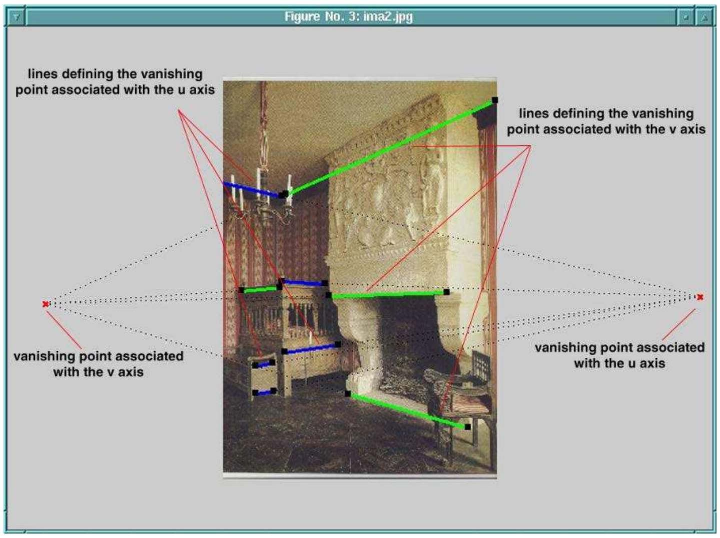

Figure 11: Determining the vanishing points

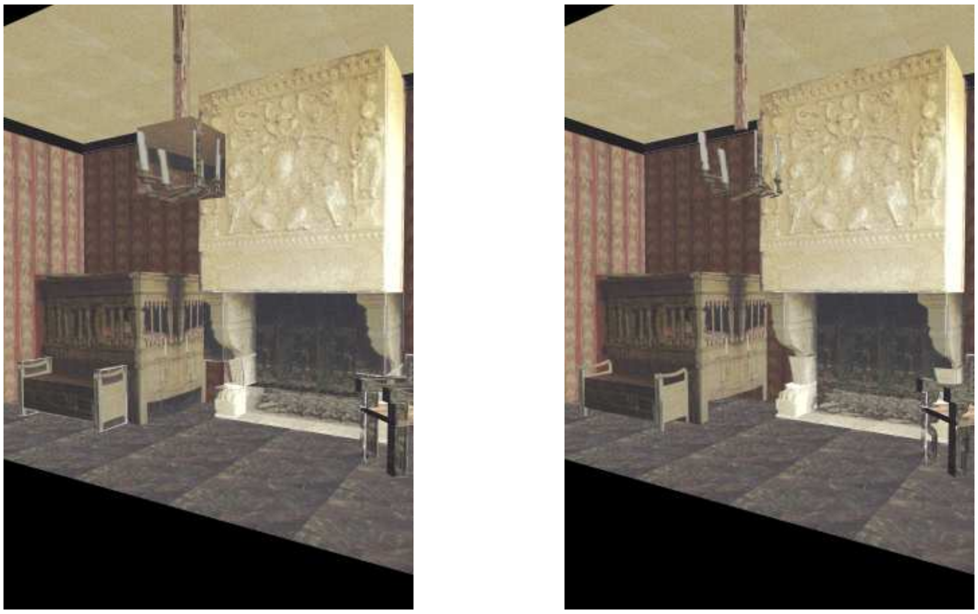

Figure 12: The reconstructed 3D model as seen from a novel viewpoint: In the left image the textures associated with the chandelier are opaque while for the right one they have been modified to make them partially transparent

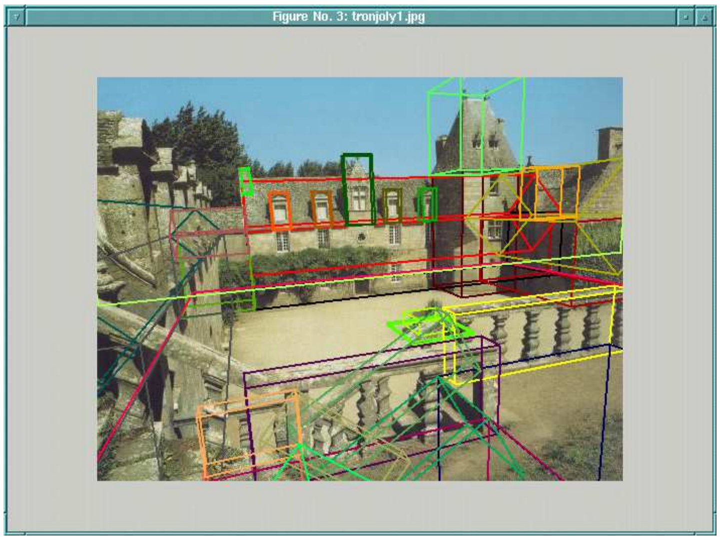

Figure 13: All the extracted 3D boxes with different colors

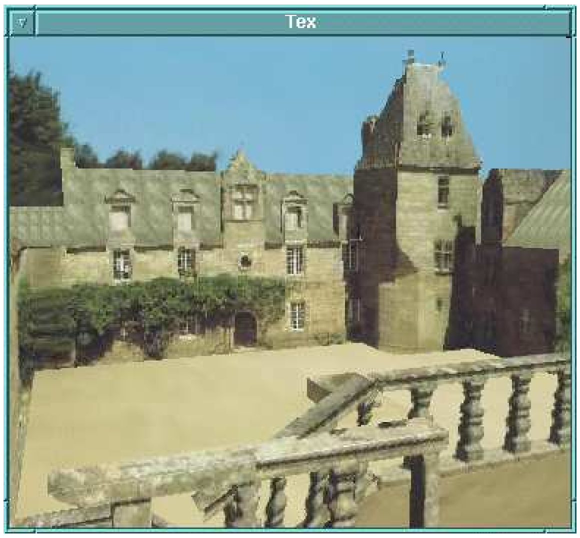

Figure 14: Reconstructed 3D model as seen from a novel viewpoint

Figure 15: Photograph with known focal length. The two sets of parallel lines are displayed in blue and green.

References

[1] L. McMillan and G. Bishop. Plenoptic modeling : An image-based rendering system. In *Computer Graphics, Annual Conference Series*, pages 39–45, 1995.

[2] S. Chen. Quicktime vr: An image-based approach to virtual environment navigation. In *SIGGRAPH*, pages 29–38, 1995.

[3] S. J. Gortler, R. Grzeszczuk, R. Szeliski, and M. F. Cohen. The lumigraph. In *SIGGRAPH*, pages 43–54, 1996.

[4] R. Szeliski. Video mosaics for virtual environments. *IEEE Computer Graphics and Applications*, pages 22–30, 1996.

[5] D. Wood, A. Finkelstein, J. Hughes, and D. Salesin C. Thayer. Multiperspective panoramas for cel animation. In *SIGGRAPH*, pages 243–250, 1997.

[6] P. Rademacher and G. Bishop. Multiple-center-of-projection images. In *SIGGRAPH*, pages 199–206, 1998.

[7] Paul E. Debevec, Camillo J. Taylor, and Jitendra Malik. Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach. In *SIGGRAPH*, pages 11–20, 1996.

[8] Paul E. Debevec, Yizhou Yu, and Georgi D. Borshukov. Efficient view-dependent image-based rendering with projective texture-mapping. In G. Drettakis and N. Max, editors, *Eurographics Workshop on Rendering*, pages 105–116. Springer Verlag, 1998.

[9] P. Poulin, M. Ouimet, and M.-C. Frasson. Interactive modeling with photogrammetry. In G. Drettakis and N. Max, editors, *Eurographics Workshop on Rendering*, pages 93–104. Springer Verlag, 1998.

[10] O. Faugeras. *Three-Dimensional Computer Vision: a Geometric Viewpoint*. The MIT Press, 1993.

[11] R. M. Haralick and L. S. Shapiro. *Computer and Robot Vision*, volume 2. Addison Wesley, Reading, Mass., 1993.

[12] R. Y. TSAI. A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf tv cameras and lens. *IEEE J. Robotics Automation*, RA-3(4):323–344, 1987.

[13] O. Faugeras and G. Toscani. Camera calibration for 3d computer vision. In *International Workshop on Machine Vision and Machine Intelligence*, pages 240–247, 1987.

[14] Youichi Horry, Ken ichi Anjyo, and Kiyoshi Arai. Tour into the picture: Using a spidery mesh interface to make animation from a single image. In *SIGGRAPH*, pages 225–232, 1997.

[15] H.Y. Shum, M. Han, and R. Szeliski. Interactive construction of 3d models from panoramic mosaics. In *IEEE conference on computer vision and pattern recognition*, pages 427–433, 1998.

[16] P. Parodi and G. Piccioli. 3d shape reconstruction by using vanishing points. *Trans. on Pattern Analysis and Machine Intelligence*, 18(2):211–217, February 1996.

[17] D. Leibowitz, A. Criminisi, and A. Zisserman. Creating architectural models from images. In R. Brunet and R. Scopigno, editors, *Eurographics*, volume 18, pages 39–50. Blackwell Publishers, 1999.

[18] Frank A. Ven Den Heuvel. 3d reconstruction from a single image using geometric constraints. *Journal of Photogrammetry and Remote Sensing, Elsevier*, 53(6):354–368, 1998.

[19] R. M. Haralick. Determining camera parameters from the perspective projection of a rectangle. *Pattern Recognition*, 22(3):225–230, 1989.

[20] B. Caprile and V. Torre. Using vanishing points for camera calibration. *International Journal of Computer Vision*, 4:127–140, 1990.

[21] L. Shigang, S. Tsuji, and M. Imai. Determining of camera rotation from vanishing points of lines on horizontal planes. *IEEE International Conference on Computer Vision*, pages 499–502, 1990.

[22] W. Chen and B. C. Jiang. 3-d camera calibration using vanishing point concept. In *CVPR*, volume 24, pages 57–67, 1991.

[23] L.-L. Wang and W.-H. Tsai. Camera calibration by vanishing lines for 3-d computer vision. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 13, pages 370–376, 1991.

[24] R. Szeliski and H.Y. Shum. Creating full view panoramic image mosaics and environment maps. In *Siggraph*, pages 251–258, 1997.

[25] B. Brillault and O'Mahony. New method for vanishing point detection. *CVGIP: Image Understanding*, 54(2):289–300, September 1991.

[26] A. Tai, J. Kittler, M. Petrou, and T. Windeatt. Vanishing points detection. In D. Hogg, editor, *British Machine Vision Conference*, pages 109–118. Springer Verlag, 1992.

[27] W. R. Mark, L. McMillan, and G. Bishop. Post-rendering 3d warping. In *12th symposium on Interactive 3D graphics*, pages 7–16. ACM, April 1997.

[28] E. Chen and L. Williams. View interpolation for image synthesis. In *Siggraph*, pages 279–288, 1993.

[29] J. Shade, S. Gortler, L. He, and R. Szeliski. Layered depth images. In *Siggraph*, pages 241–242, 1998.