EG UK Theory and Practice of Computer Graphics (2006)
M. McDerby, L. Lever (Editors)

ATIP: A Tool for 3D Navigation inside a Single Image with Automatic Camera Calibration

K. Boulanger, K. Bouatouch and S. Pattanaik

IRISA, Université de Rennes I, France
University of Central Florida, United States

Abstract

Automatic Tour Into the Picture (ATIP) is an extension of the Tour Into the Picture method [HAA97] that allows an approximative but visually convincing 3D walk-through inside a single image by rendering a box textured using the input image data. The original algorithm requires a long and tedious user interaction to determine the box dimensions and the perspective parameters, and imposes several constraints on the input image orientation. The goal of this paper is to present a framework providing fully automatic and fast camera calibration for any view orientation without using a calibration target. Our method reduces the user interaction, hence only a couple of seconds are required between the input image loading and the final walk-through.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Image-Based Rendering, I.4.1 [Image Processing and Computer Vision]: Camera calibration

1. Introduction

Image-Based Modeling (IBM) aims at recovering the geometry and the photometry of objects from one or more images. Tour Into the Picture (TIP), developed by Horry et al. in [HAA97] and extended in [Anj99, Chu01, YCHY03, CSS04], is an image-based modeling method that coarsely models the scene of a single input image as a texture mapped box. The approximation is coarse since real scenes cannot be modeled using a single box. However, this box and the recovered textures for its faces allow an user to carry out a visually pleasant walk-through inside the 3D scene. The generated data are very compact (a maximum of 5 compressed textures and geometry data for a simple box) and the rendering of a scene is inexpensive (a maximum of 5 texture mapped quadrilaterals). Therefore this system is well-suited for a navigation with photographic quality on Personal Digital Assistants (PDAs) and personal computers.

The main drawback of this method is the long and tedious user interaction during the camera calibration step since every parameter has to be defined manually. Additionally, several constraints are imposed on the input image, particularly the requirement that a single vanishing point be present in the image, and the horizontal and vertical orientation of the captured scene be exactly preserved in the image. It is almost impossible to have a picture with horizontal and vertical lines perfectly aligned with the borders of an image captured using a hand-held camera. Our observation is that, in this case, the rotation around the view axis is about 1 to 5 degrees. The manual fitting with the TIP algorithm becomes very approximative in this case.

The objective of this paper is to fully automate the camera calibration step of the Tour Into the Picture algorithm and minimize the constraints on the input image. We call our method *Automatic Tour Into the Picture* (*ATIP*).

We use a vanishing point based approach for the calibration operation. Several papers have been proposed for camera calibration using vanishing points, however some constraints have to be satisfied. For example, the camera has to be hold upright. A common method is the use of a calibration target in the input image as in [CJ91, WT91]. Some approaches make use of lines present in the original image, typically in architectural scenes. These lines can be selected manually as in [GMMB00, CSS04] or can be retrieved using a line detection algorithm, usually a Hough transform [Rot02, LMLK94, CLPS01]. From the intersection of these lines, vanishing points can be determined using clustering algorithms [BBV00, LJN02, Rot02], which group lines contributing to one of the vanishing point together, or using

projection based methods that project lines to another space to find their intersection, as performed using the *gaussian sphere projection* [Bar83,LMLK94,CLPS01,LJN02]. These methods have common drawbacks: precision problems for the vanishing point coordinates and computationally expensive processing.

One of the main contributions of this paper is the automatic calibration of the camera (focal length and rotation matrix) from a single image, in a few seconds and with a good precision. Another main contribution is to reduce the constraints on the input image. The input image may be easily acquired since no calibration target is needed and no alignment with the scene is required, therefore there is no need for a tripod for the camera. Our calibration method removes the main constraints of the previously mentioned papers.

The structure of the paper is as follows. We start by giving the outline of our algorithm. Next, we give details about the main three steps of our algorithm: vanishing points detection, camera calibration and coarse 3D reconstruction. We then give some implementation hints and results. We finish by giving some ideas of future work.

2. Outline of our algorithm

In this section, we present ATIP, the automation of the TIP method. The outline of our algorithm is given in Figure 1. We adapt various methods from the field of computer vision to process the input image for automatic camera calibration. We decompose camera calibration into two parts: automatic vanishing points detection (Section 3) and automatic camera parameters extraction (Section 4). Contrary to the original TIP algorithm that requires the presence of a single vanishing point, we manage the cases where one, two or three vanishing points are present in the input image, and thus allow any orientation of the camera used to capture the input image.

Vanishing point detection begins with a simple edge detection step on the input image. Following this step, a dominant lines detection algorithm is executed using the coordinates of the edge points. We use a polar Hough transform described in Section 3.1 to perform this detection. We assume that these detected lines include most of the vanishing lines of the image. Then, we find the intersection points of the lines to estimate the coordinates of the vanishing points. Using a method described in Section 3.2, we project these lines onto a hemisphere to detect vanishing points that are near and far from the image center. Camera parameters such as focal length and rotation are determined using these points, as explained in Section 4. The geometry of the box fitting the scene is determined using the camera parameters and minimal user interaction. Note that this is the only step of our algorithm requiring such an interaction. Section 5 details the computation of the textures for the faces of the box,

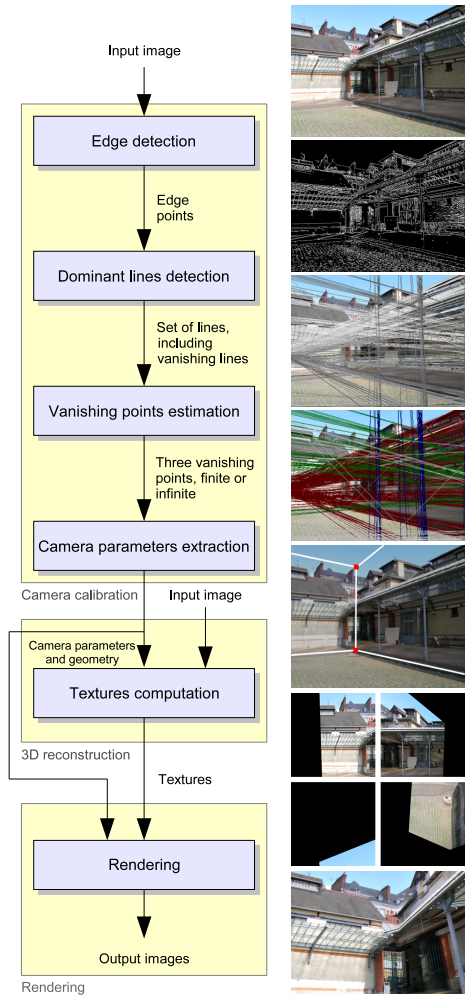

Figure 1: *Summary of our algorithm, from the input image to the final 3D scene used for navigation.*

computed using the input image and the camera parameters. 3D walk-through in the captured scene is then possible, by changing the position and rotation of the camera and rendering the textured box.

3. Vanishing points detection

To calibrate a camera given a single input image with no additional information, perspective projection parameters have to be extracted from this image. One way to achieve this goal is to use vanishing points. Perspective projection of parallel lines in world space creates *vanishing lines* in image space that intersect in a point called *vanishing point*. If the parallel lines are parallel to the image plane, the vanishing lines are also parallel to this plane and the vanishing point is at an infinite distance from the image center, hence is called *infinite vanishing point*. Otherwise it is a *finite vanishing point*.

A line in world space starting from the optical center of the camera and intersecting a vanishing point in image space gives the direction of the associated parallel lines.

Our camera calibration is based on the following assumption: the main vanishing lines in the input image correspond to three orthogonal directions in world space. This assumption works for many types of scenes, in particular indoor or outdoor architectural scenes. Each of these orthogonal directions is associated with a finite or infinite vanishing point. Different combinations of such vanishing points create three different configurations. Rotation around the view direction due to the tilting of the camera has no effect on these configurations. We detail the processing of these three configurations in Section 4.

The vanishing lines mostly appear as edges in the input image. Hence the first step of our vanishing line detection algorithm is an edge detection for which we use a classic Laplacian based method [FP02].

3.1. Dominant lines detection

The edge detection step results in a set of points defining the contours of objects within the image. From these contours, we extract a set of straight lines using a polar Hough transform [FP02]. The vanishing lines belong to this set. A line on the image plane is parameterized using an angle θ and the distance of the line from the origin ρ, corresponding to a point (ρ, θ) after transformation. An edge point on the image plane is considered as the intersection of an infinity of these lines. Therefore we transform each edge point using the *1 to m* Hough transform to a curve in Hough space defined by the equation below [CLPS01]:

$$\rho = x\ \cos(\theta) + y\ \sin(\theta) \tag{1}$$

We show in Section 6 how we implement efficiently this transform. When processing a very large number of input points, the cost of this method is low relatively to an *m to 1* Hough transform [FP02] ($O(n)$ rather than $O(n^2)$). The result of the transform is a 2D grid of accumulators, for which the values are high where a high number of curves are intersecting, corresponding to a large number of aligned edge points. A simple thresholding could be performed to identify these points, however the results are very noisy (Figure 2*(f)*) because of large groups of adjacent cells with high value (Figure 2*(e)*), often due to highly textured zones in the input image. We want to detect peaks (Figure 2*(a)*), which are local high values. So we apply a high-pass filter to isolate these peaks (Figures 2*(c)* and 2*(g)*), followed by the thresholding (Figures 2*(d)* and 2*(h)*). The threshold can be manually set but the default value we use works for most images. As shown in section 6, the high-pass filter allows detection of noisy lines as in forest scenes and hand-drawn images.

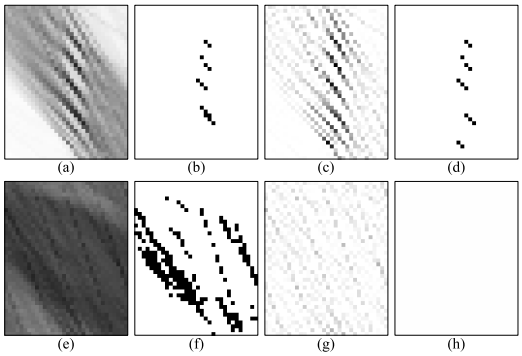

Figure 2: *Application of a high-pass filter on the Hough transform to keep the peaks. The first row is a case where peaks have to be detected. The result with and without filtering are similar. The second row is a noisy zone, the filter is very efficient in this case.*

3.2. Vanishing points estimation

To be able to calibrate the camera for the reconstruction of the 3D scene, we need three vanishing points, finite or infinite. From the previous step, we obtain a set of lines that includes vanishing lines. The intersection of the vanishing lines should give the vanishing points. However, they are not precise points but *zones of intersections*. Moreover, these intersections can be outside the image. If a vanishing point is far enough from the image center, we consider it as infinite.

A classic method to detect the vanishing points is the *gaussian sphere projection* [Bar83, LMLK94, CLPS01, LJN02, Rot02, BBV00], also known as *gnomonic projection*. This method projects lines from image space onto a sphere that is tangent to the image plane at the center of the image (Figure 3*(a)*). The projection of lines are circles around the sphere that is discretized to perform an accumulation algorithm. The cells with the highest accumulation values correspond to the intersection points of the circles. Then the center of these cells are projected back to the image plane to obtain the vanishing point coordinates (Figure 3*(b)*). The cells on the equator correspond to infinite vanishing points. The detection of such points is the main advantage of this projection.

In our approach, we use a modified version of the gaussian sphere projection method, which is more precise and faster. Firstly, a sphere is used to perform the projection. Since the lower half of the sphere is a symmetrical version of the upper half, we use the upper hemisphere only. Therefore, the projection and the accumulation steps are faster and the intersections to be detected are present only once. We set the hemisphere diameter to the shortest side length of the image (Figure 3*(a)*). We propose a subdivision method that consists of a grid whose resolution progressively decreases with the distance from the equator while keeping a constant precision

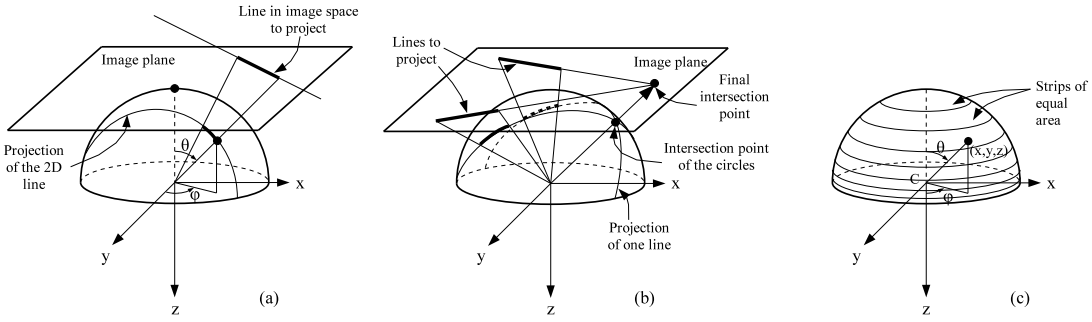

Figure 3: (a) *Projection of a vanishing line onto a unit hemisphere,* (b) *intersection of two curves on the hemisphere to find the intersection of the corresponding lines in image space,* (c) *strips of equal area allowing equal area accumulator cells, hence a better precision for vanishing point coordinates in image space.*

of the vanishing point direction from the image center. We define such constraints to achieve a better distribution of the hemisphere cell projections onto the image plane. One possible way to achieve this goal is to use equal area subdivision of the hemisphere by modifying the subdivision of θ, unlike in [Bar83, CLPS01, LJN02, LMLK94] where the subdivision of θ is always uniform.

First, φ is uniformly discretized into N different angles:

$$\varphi_i = \frac{i}{N} \cdot 2\pi \qquad i \in \{0, \ldots, N-1\} \tag{2}$$

We want a subdivision with accumulators of equal area. Since φ is uniformly discretized, each strip on the hemisphere for a given θ range must have an equal area (Figure 3*(c)*).We consider a unit sphere, so the image plane has coordinates for the height axis in $[-1,1]$. To obtain uniform strip areas, we define an angle ψ_j, depending on θ_j, which is proportional to the area of the portion of sphere from the pole to the angle θ_j. This area is equal to

$$A(\theta_j) = \int_0^{\theta_j} \int_0^{2\pi} \sin\theta \, d\varphi d\theta = 2\pi(1-\cos\theta_j) \tag{3}$$

So we can define

$$\psi(\theta) = \frac{\pi}{2}(1-\cos\theta) \in [0, \pi/2] \qquad \theta \in [0, \pi/2] \tag{4}$$

The area of the portion of sphere from the pole to the angle θ_j is then proportional to $\psi(\theta_j)$:

$$A(\theta_j) = 4\psi(\theta_j) = 4\psi_j \tag{5}$$

Projecting the dominant lines in the image consists in finding the equation of the corresponding curves. These ones are determined by computing the intersection between the plane P (defined by the line to project and the hemisphere center) and the hemisphere. We obtain the following set of equations to solve, where $(n_x, n_y, n_z)^T$ is the normal to the plane P with $n_z \geq 0$ as a convention, (x,y,z) is any point on the unit hemisphere:

$$\begin{cases} n_x\, x + n_y\, y + n_z\, z = 0 \\ x^2 + y^2 + z^2 = 1 \\ z \leq 0 \end{cases} \tag{6}$$

Using the equations allowing to convert cartesian coordinates to spherical coordinates (θ is on the negative side of the $\vec{z}$ axis), we solve the equations system (6) and we obtain the following relation for the projected curve $C(\varphi)$, giving the elevation angle θ depending on the azimuthal angle $\varphi \in [0, 2\pi]$:

$$\theta = C(\varphi) = \arcsin \frac{n_z}{\sqrt{n_z^2 + (n_x \cos\varphi + n_y \sin\varphi)^2}} \tag{7}$$

Using the function giving the angle ψ depending on the elevation angle θ and the formula $\cos(\arcsin(x)) = \sqrt{1-x^2}$ for $x \in [0,1]$, we obtain, using discretized angles:

$$\psi(C(\varphi_i)) = \frac{\pi}{2}\left(1 - \frac{n_x \cos\varphi_i + n_y \sin\varphi_i}{\sqrt{n_z^2 + (n_x \cos\varphi_i + n_y \sin\varphi_i)^2}}\right) \tag{8}$$

For simplification purpose, we swap the definition of the elevation angle θ, 0 corresponding to the equator and $\pi/2$ to the pole. The final equation of the curves to be drawn in the accumulation space are

$$\psi(C(\varphi_i)) = \frac{\pi}{2} \frac{\alpha(\varphi_i)}{\sqrt{n_z^2 + \alpha(\varphi_i)^2}} \tag{9}$$

with $\alpha(\varphi_i) = n_x \cos\varphi_i + n_y \sin\varphi_i$ and φ_i defined as in Equation 2.

The projection space, parameterized by φ and $\psi(C(\varphi))$, is a uniform grid. Note that discretizing φ and $\psi(C(\varphi))$ uniformly amounts to sample φ and θ so that the accumulation areas of the hemisphere are of equal size. The Figure 4 shows an example of lines projected as curves onto this reparameterized accumulation space.

The detection of the maxima on the accumulation space is

not really easy due to the presence of noise. We use a low-pass filter to minimize it. We want to find three accumulation cells that correspond to the three vanishing points. The trivial approach is to detect the three accumulation cells with the maximum value. This approach does not work since each of these three cells often belong to a single zone of cells corresponding to one vanishing point. The algorithm we use instead is the following:

do 3 times
{
1. Apply a low-pass filter to the accumulator cells,
2. Look for the cell with the maximum value,
3. Project this cell back to the image plane to get the coordinates of a vanishing point,
4. Look for the dominant lines that are vanishing lines associated with the current vanishing point,
5. In the unfiltered accumulation space, redraw into the accumulation space the curves corresponding to this set of lines while decrementing the accumulator values rather than incrementing them

}

We use a simple thresholding with the elevation angle θ to determine whether a vanishing point is finite or infinite.

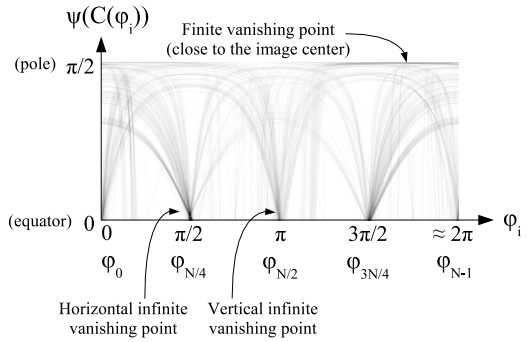

Figure 4: *Result of the accumulation of 2164 curves onto the hemisphere. The spherical coordinates are mapped to a 2D grid for visualization (*800 *by* 600 *cells). The darker are the points, the higher are their accumulation value.*

To find the dominant lines corresponding to a finite vanishing point, we use the distance from the point to the line in image space as a measure. All the lines whose distance is below a certain threshold (say a few pixels) are chosen as the contributors. A threshold distance relative to the input image size is more robust than that specified in a fixed number of pixels so is useful in input images of arbitrary resolution. For an infinite vanishing point, the angle between the vanishing point direction from the image center and the line to test in image space is a criterion that works well (the threshold we define is equal to a few degrees).

Given this set of three vanishing points, the next step of our algorithm is to carry out the camera parameters extraction.

4. Camera parameters extraction

Our goal is to compute the intrinsic and extrinsic parameters of the camera. Given the coordinates of three vanishing points, the focal length (intrinsic parameter) and the rotation matrix (extrinsic parameter) are the only parameters that can be retrieved. We make two assumptions: the principal point P of the camera is set to the center of the image plane and the pixels of the image are squares. The rotation matrix $(\vec{u}\ \ \vec{v}\ \ \vec{w})$ transforms points from world space to camera space. Its columns are the vectors of the world coordinates frame expressed in camera space. In Section 3, we assumed that the directions of the three vanishing points from the optical center of the camera are orthogonal. Hence the following set of relations must hold true for the final calibrated camera:

$$\begin{cases} f > 0 \\ \vec{u}\cdot\vec{v} = \vec{v}\cdot\vec{w} = \vec{w}\cdot\vec{u} = 0 \\ ||\vec{u}|| = ||\vec{v}|| = ||\vec{w}|| = 1 \end{cases} \tag{10}$$

In Section 3, we discussed about three different finite and infinite vanishing point configurations. Processing for each of them is different and is detailed below.

4.1. One finite vanishing point, two infinite vanishing points

This situation occurs when two axes of the world coordinate frame are parallel to the image plane. The Figure 5*(a)* shows the terms involved for the calibration of the camera using one finite vanishing point $\overrightarrow{OV_p} = (vp_x, vp_y, -f)^T$ and two infinite vanishing points of directions $\overrightarrow{I_1} = (I_{1x}, I_{1y}, 0)^T$ and $\overrightarrow{I_2} = (I_{2x}, I_{2y}, 0)^T$. It is not possible to compute a focal length from a single finite vanishing point. For this reason, in [Anj99], the focal length of the camera is set manually. It is possible to make an estimate of this value using the additional information given by the infinite vanishing points.

First $\vec{w'}$, a non-normalized form of the coordinate axis $\vec{w}$ is computed from the finite vanishing point:

$$\vec{w'} = (w'_x, w'_y, w'_z)^T = -\overrightarrow{OV_p} = (-vp_x, -vp_y, f)^T \tag{11}$$

The coordinate axis $\vec{u}$, as indicated in Figure 5*(b)*, lies on the plane defined by the points O, P and the direction $\overrightarrow{I_1} = (I_{1x}, I_{1y}, 0)^T$. Then $\vec{u'}$, the non-normalized version of $\vec{u}$, can be expressed as $\vec{u'} = (I_{1x}, I_{1y}, u'_z)^T$. We want that $\vec{u'}$ and $\vec{w'}$ belong to an orthogonal coordinate frame, so $\vec{u'}\cdot\vec{w'} = I_{1x}w'_x + I_{1y}w'_y + u'_z w'_z = 0$. We obtain

$$u'_z = \frac{I_{1x}vp_x + I_{1y}vp_y}{f} \tag{12}$$

The vector $\vec{v'}$ is determined exactly the same way (Figure

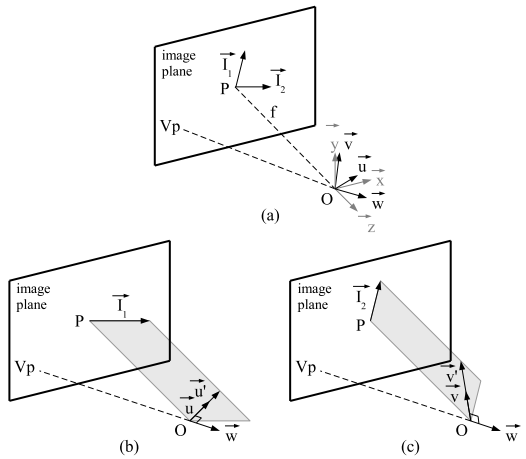

Figure 5: *Camera calibration with one finite and two infinite vanishing points.* (a) $(\vec{x},\vec{y},\vec{z})$ *is the camera coordinate frame,* $(\vec{u},\vec{v},\vec{w})$ *the world coordinate frame to be found,* V_p *the finite vanishing point,* $\vec{I_1}$ *and* $\vec{I_2}$ *the infinite vanishing point directions. P is the principal point.* (b) *Determination of* $\vec{v}$ *and* $\vec{w}$ *depending on the infinite vanishing points.*

5(c)), $\vec{v'} = (I_{2x}, I_{2y}, v'_z)^T$ with

$$v'_z = \frac{I_{2x}vp_x + I_{2y}vp_y}{f} \tag{13}$$

To obtain an orthogonal coordinate frame with $\vec{u'}$, $\vec{v'}$ and $\vec{w'}$, the relation $\vec{u'} \cdot \vec{v'} = 0$ has also to be valid. We insert the Equations 12 and 13 into this relation and we obtain the focal length (positive):

$$f = \sqrt{\left| \frac{(I_{1x}vp_x + I_{1y}vp_y)(I_{2x}vp_x + I_{2y}vp_y)}{I_{1x}I_{2x} + I_{1y}I_{2y}} \right|} \tag{14}$$

In this equation, the denominator is same as $\overrightarrow{I_1} \cdot \overrightarrow{I_2}$. These vectors are almost orthogonal, so the denominator is close to zero, reducing considerably the accuracy of the focal length estimate. However, results are accurate enough when V_p is far from the principal point, which is the image center. Half of the image size is enough to obtain correct results.

Because of the inaccuracy involved in this computation, the user may choose to set the focal length manually. Focal length corresponding to 48 degrees vertical field of view is a good approximation for most images. This setting is generally within 10% error of the real value. This error is not really visible in the final rendered scene since this parameter influences only the distortion of the depth due to the perspective effect. In situation where the focal length is chosen manually, the vectors $\vec{u'}$, $\vec{v'}$ and $\vec{w'}$ have to be computed using Equations 11, 12 and 13. Note that it does not guarantee the orthogonality between $\vec{u'}$ and $\vec{v'}$, and must be corrected.

Typically the vertical lines are abundant and more precise in many images, so we retain $\vec{v'}$ and $\vec{w'}$ and compute $\vec{u'}$ using a cross product, $\vec{u'} = \vec{v'} \times \vec{w'}$. The rotation matrix of the camera $(\vec{u}\ \ \vec{v}\ \ \vec{w})$ that respects the conditions of Equations 10 is obtained by normalizing $\vec{u'}$, $\vec{v'}$ and $\vec{w'}$.

4.2. Two finite vanishing points, one infinite vanishing point

This situation is illustrated in Figure 6 and happens when one of the three axes of the world coordinate frame is parallel to the image plane. A method to compute the focal length and the rotation matrix from the coordinates of the vanishing points is given in [GMMB00]. However, we found an easier way to obtain the same results.

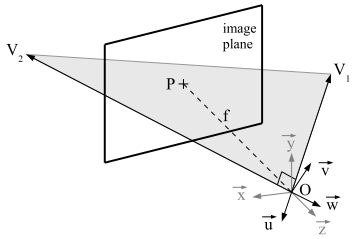

Figure 6: *Camera calibration with two finite and one infinite vanishing points.*

Two finite vanishing points, V_1 and V_2, allow to define the vectors $\overrightarrow{OV_1}$ and $\overrightarrow{OV_2}$ as directions of two of the axes of the world coordinate frame. The third vanishing point that is considered as infinite is actually a finite vanishing point that is very far from the image center. If the line (V_1V_2) passes through the point P, then the third vanishing point is really infinite since the triangle (OV_1V_2) is orthogonal to the image plane, hence $\vec{v}$ is parallel to the image plane. As the precision of the two finite vanishing points coordinates is better than that of the infinite one, we use them to determine the directions of two axes of the world coordinate frame and compute the direction of the third axis while ensuring the orthogonality of the rotation matrix.

We consider that the 3D directions corresponding to the vanishing lines are orthogonal. So it is easy to find the focal length. The vectors $\overrightarrow{OV_1} = (v_{1x}, v_{1y}, -f)^T$ and $\overrightarrow{OV_2} = (v_{2x}, v_{2y}, -f)^T$ have to be orthogonal, so $\overrightarrow{OV_1} \cdot \overrightarrow{OV_2} = 0$ and finally, with a positive focal length:

$$f = \sqrt{|v_{1x}v_{2x} + v_{1y}v_{2y}|} \tag{15}$$

Once the focal length is known, the rotation matrix $(\vec{u}\ \ \vec{v}\ \ \vec{w})$ that respects the conditions of Equations 10 is simply obtained by:

$$\vec{u} = -\frac{\overrightarrow{OV_1}}{\|\overrightarrow{OV_1}\|} \qquad \vec{w} = -\frac{\overrightarrow{OV_2}}{\|\overrightarrow{OV_2}\|} \qquad \vec{v} = \vec{w} \times \vec{u} \tag{16}$$

4.3. Three finite vanishing points

This situation occurs when no axis of the world coordinate frame is aligned with the image plane. Calibration of the camera using three finite vanishing points is an over-constrained problem. The classic method to solve it is analytic fitting, typically using a SVD as in [CDR99]. We decided to use a simpler method that gives good results.

We assume that the directions represented by the three vanishing points are orthogonal. However, due to precision problems, no set of three orthogonal directions corresponding to the extracted vanishing point coordinates can be found. Therefore we use two of the three vanishing points to compute the focal length as in the previous section, and compute the third direction using the cross product of the two first directions. Three different couples of vanishing points can be chosen, (V_1,V_2), (V_2,V_3) and (V_3,V_1). The couple that gives the third vector closest to the third vanishing point direction is chosen (maximum of their dot product).

Our method gives a coordinate frame that is aligned exactly with two vanishing point directions. The SVD method should give a coordinate frame that is unaligned with the three vanishing point directions but with a lower error. We believe that our method should give better visual results, even if the global error is higher, since we are visually sensitive to coordinate frames that are not aligned with the content of the input image.

5. Coarse 3D reconstruction

As for the original Tour Into the Picture algorithm [HAA97], we use a box to coarsely reconstruct the 3D scene. In our approach, this box is aligned with the world axes $(\vec{u},\vec{v},\vec{w})$. We also know the focal length that allows to compute perspective corrected textures mapped to the faces of the box. The only unknown parameters at this point are the size and position of the box. From a single image, these information can only be approximated and the algorithm should *interpret* the content of the input image to determine these parameters. It is actually faster and simpler to ask the user to give these information the simplest possible way. We use an interactive interface, named *scene editor*, to input some of these information.

The box is rendered on the top of the image in wireframe, red points are displayed on its visible corners and can be interactively moved by the user (fifth image of Figure 1). Since the virtual camera is calibrated, the user always sees the box with its appropriate perspective distortion. This user interaction is easy and fast, just a few seconds are necessary to correctly move the box corners. Only this step of the algorithm requires such an interaction.

The last automatic step is the computation of the texture data for the faces of the reconstructed box. Each face of the box is assigned a texture and divided into a regular grid representing *texels*. To know the color of each texel, we determine the line going from the camera optical center to the center (x,y,z) of the current texel and compute the coordinates of its intersection with the image plane. The coordinates are then converted into image space (x_i,y_i) to get the color from the input image. Bilinear interpolation of the input image is used to improve the texture quality. If the intersection is out of the bounds of the input image, a black color is assigned to the texel.

The resolution of the face textures is determined by two parameters. The first one is the quality required for the final application. It is set by the user based on the final display resolution and the type of platform (for example, PDAs require less resolution). The second parameter is the area of projection of the associated box face onto the input image plane. A set of thresholds is used to determine the resolution of the texture depending on this area. The ratio between the projected width and the projected height of the texture has also an influence on the final resolution.

6. Implementation and Results

The software we designed is made of two modules: *ATIP Maker* and *ATIP Navigator*. The first module performs all the steps described from Sections 3 to 5, from a 2D input image to a file containing camera, geometry and textures information. The second module is a navigator allowing to walk through the reconstructed 3D scene. A few frames from the navigation into the scene of an example image is given in Figure 7. Since the rendering is extremely simple, the rasterization of no more than 5 textured quadrilaterals, the navigator can be ported to PDAs. Our *ATIP Navigator* runs both on *PC* and *Pocket PC* (Figure 8). On the latter platform, using software rasterization on a 240×320 screen, we obtained 3D navigation at 13 frames per second using bilinear filtering for the textures and 26 frames per second without filtering.

Figure 7: *An input image (upper left corner) and its rendering from three different viewpoints.*

Our approach allows us to introduce a few optimization

Figure 8: *ATIP navigator running on a Pocket PC.*

into our algorithm. The first optimization we made is the resizing of the original image. By shrinking it, the result of the camera calibration is less precise but a lot faster. The polar Hough transform, described in Section 3.1, is the longest step and is proportional to the number of edge points to be transformed. In our implementation the user is allowed to choose to work on the normal, half and quarter size input image. For the quarter size input, we observed speed-ups up to 25, more than the expected speed-up of 16 times. This ratio comes from the edge detection step that does not return a number of points proportional to the image area. Quarter size image gives precise enough results for most of the input images. The second optimization we made is to use fixed point arithmetic for computation involving the polar Hough transform (Section 3.1). The chosen number of bits for the integer and the fractional parts gives enough precision for the representation of ρ. The cosine and sine functions used in Equation 1 are stored as fixed point numbers in tables. The use of look-up tables rather than direct function calls does not imply any loss of precision since the sampling of θ is the same for each edge point. The resulting speed-up is about 15. We finally achieved speed-ups up to $25 \times 15 = 375$.

The following table gives the calibration times for a 5 megapixels image (Figure 1) on a *1.8 GHz Pentium-M PC*. The *fast* and *very fast* modes respectively correspond to the processing with half and quarter resolution of the input image. The number of edge points transformed into curves for the *fast* mode is 193,544.

mode	camera calibration time (seconds)	speed-up	relative error
normal	19.4		0.6%
fast	3.38	5.74	2.17%
very fast	0.97	20	3.79%

The relative error for the focal length in the last column is calculated based on the value returned by the camera in a text file. This error is actually not visible in the final rendered scene since a slightly wrong focal length has only a small distortion in perspective effect. Moreover, it has no influence on the rotation matrix.

The texture computation time depends on the texture resolution. For textures with a maximum resolution of 256 pixels (for PDAs) and 1024 pixels, computation times are 0.25 second and 1.17 second respectively.

Figure 9: *Three difficult scenes for the camera calibration: non-architectural scene, scene with strong presence of curves and hand-drawn image. The input images are in the first column and a rendering from another viewpoint in the second one.*

More than 75% of the two dozens images we have experimented with have been successfully calibrated and then used for 3D reconstruction. Half of them have been processed with the *very fast* mode while the other half has been processed with the *fast* mode. For most of the tested images, the default parameter setting works well (thresholds used by the different filters involved, threshold for the polar Hough transform, distance and angle to consider dominant lines as contributing to a vanishing point, etc.). Sometimes, minor changes are required. Since our method provides results very quickly, it is relatively easy to change these thresholds and get the desired results.

Usually, vanishing point detection algorithms are applicable only to architectural scenes since they contain many straight lines, some of them being orthogonal to others. Using a high-pass filter in the Hough space (Section 3.1), our algorithm is able to detect vanishing points from input images containing noisy lines. Figure 9 shows an example of this situation with a forest scene and a building containing many curved lines that are selected by the dominant line detection step. In these cases, the values of the used thresholds have just to be chosen in a less strict way. The last row of the Figure 9 shows the results of the detection using a hand-

drawn image without using a ruler. The lines are not straight and the zones of vanishing lines intersections are large. Our algorithm is able to easily detect the vanishing points.

7. Conclusion and Future Work

Automatic Tour Into the Picture is an extension of the Tour Into the Picture algorithm [HAA97] in that it simplifies the intervention of the user. Many constraints are removed, particularly those related to the camera orientation. The user just takes a shot, uses it as input data and runs our algorithm to obtain a coarsely reconstructed 3D scene and create photographic quality images from different camera positions and orientations in a few seconds. The user interaction is restricted to its minimum. The camera calibration is robust enough to work with non-architectural scenes such as forests. It is also fast enough to process several input images of the same 3D environment when the aim is to navigate through this environment. In this case, a unique scale is needed for all the reconstructed 3D models, each one being associated with one input image. As the size of the generated data is small and the rendering is fast, our method can be embedded into hand-held devices such as PDAs or cell phones, and used in applications such as navigation systems for pedestrians.

In the original paper presenting Tour Into the Picture [HAA97], the foreground items are determined using an alpha matte drawn manually by the user, then these items are rendered using billboards. We could follow the same approach in our algorithm but it would require an intensive user interaction. An extension to our work will be to add automatic foreground element detection and background filling.

We defined a camera calibration method that recovers automatically the focal length and the rotation matrix of the camera. This information can be used for purposes other than Tour Into the Picture, for example for manual reconstruction of 3D scene from a single image as done by Guillou et al. in [GMMB00].

8. Acknowledgements

The authors are grateful to Eric Marchand and Murat Balci for their pertinent remarks on this work.

References

[Anj99] ANJYO K.: "Tour Into the Picture" as a non-photorealistic animation. *Computer Graphics 33-1* (1999), 54–55.

[Bar83] BARNARD S. T.: Interpreting perspective images. *Artificial Intelligence 21* (1983), 435–462.

[BBV00] BRÄUER-BURCHARDT C., VOSS K.: Robust vanishing point determination in noisy images. In *International Conference on Pattern Recognition (ICPR'00)* (September 2000), vol. 1, pp. 1559–1562.

[CDR99] CIPOLLA R., DRUMMOND T., ROBERTSON D.: Camera calibration from vanishing points in images of architectural scenes. In *BMVC99* (1999).

[Chu01] CHU S.-H.: *Animating Chinese Landscape Paintings and Panoramas*. Master's thesis, Hong Kong University of Science and Technology, August 2001.

[CJ91] CHEN W., JIANG B. C.: 3-D camera calibration using vanishing point concept. *Pattern Recognition 24*, 1 (1991), 57–67.

[CLPS01] CANTONI V., LOMBARDI L., PORTA M., SICARD N.: Vanishing point detection: representation analysis and new approaches. In *CIAP01* (2001), pp. 90–94.

[CSS04] CAO Z., SUN X., SHI J.: Tour into the picture using relative depth calculation. In *VRCAI '04: Proceedings of the 2004 ACM SIGGRAPH international conference on Virtual Reality continuum and its applications in industry* (2004), pp. 38–44.

[FP02] FORSYTH D., PONCE J.: *Computer Vision: A Modern Approach*. Prentice Hall, 2002.

[GMMB00] GUILLOU E., MENEVEAUX D., MAISEL E., BOUATOUCH K.: Using vanishing points for camera calibration and coarse 3D reconstruction from a single image. *The Visual Computer 16-7* (2000), 396–410.

[HAA97] HORRY Y., ANJYO K.-I., ARAI K.: Tour Into the Picture: using a spidery mesh interface to make animation from a single image. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques* (1997), pp. 225–232.

[LJN02] LEE S. C., JUNG S. K., NEVATIA R.: Automatic integration of facade textures into 3D building models with a projective geometry based line clustering. *Computer Graphics Forum 21*, 3 (September 2002).

[LMLK94] LUTTON E., MAITRE H., LOPEZ-KRAHE J.: Contribution to the determination of vanishing points using Hough transform. *IEEE Transactions on Pattern Analysis and Machine Intelligence 16-4* (April 1994), 430–438.

[Rot02] ROTHER C.: A new approach for vanishing point detection in architectural environments. *IVC 20*, 9-10 (2002), 647–656.

[WT91] WANG L.-L., TSAI W.-H.: Camera calibration by vanishing lines for 3-D computer vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence 13*, 4 (1991), 370–376.

[YCHY03] YOON S., CHEN H.-J., HSU T., YOON I.: Web-based virtual tour using the Tour Into the Picture (TIP) technique. In *9th International Conference on Distributed Multimedia Systems* (September 2003), pp. 105–108.