Real-time Registration for Image Mosaicing

E. Noirfalise, J.T. Lapresté, F. Jurie and M Dhome
LASMEA - CNRS UMR 6602, Université Blaise-Pascal,
F-63177 Aubière - FRANCE
jurie@lasmea.univ-bpclermont.fr

Abstract

This paper presents a real-time tracking technique for constructing *spherical image mosaics* from sequences of images. The main contribution is a tracking algorithm able to estimate in real-time camera rotations during the sequence, assuming that the optical center of the camera remain fixed. This assumption allows us to consider a sphere centered at this particular point and having its radius equal to the focal length of the camera. The problem is to estimate for each image of the sequence the position of the image plane relatively to this sphere and to back project the image to create the global spherical mosaic image. The proposed algorithm allows, during a first stage, to track in real time (less than 12ms for each image) the camera rotations around the optical center. This is done by using a real-time template matching algorithm, using the formalism described in [2], adapted in order to estimate only 3D camera rotations. After this stage, each one of the images of the sequence are located relatively to the spherical image. In a second step, the spherical image is build by back projecting on the sphere the images of the sequence. This paper is mainly devoted to the description of the first stage. Experimental results proving the efficiency of the proposed approach are given.

1 Introduction

The automatic construction of large image mosaics is an active area of research in computer vision and computer graphics. It can be used for various applications like aerial and satellite photographs, stabilization, compression, *etc*, as explained by Shum and Szeliski [5]. This article deals with image mosaics for the emulation of film-based panoramic photography for applications such as the construction of virtual environments and virtual travels. Numerous techniques have been used to obtain panoramic images of real world scenes like using a long strip, a panoramic camera, fish eye lens or parabolic mirrors[4, 7].

Software based methods can be substituted for the previously mentioned hardware-intensive methods. They consists in aligning regular photographic or video images in order to cover the whole viewing space, using a *stitching* algorithm [6, 1].

In this article we propose an approach allowing to have complete (full view) panoramas which cover the whole viewing sphere. By this way, the user can look at any direction. The goal of our work is to provide an interactive system allowing to “paint “ a viewing sphere in real time with a simple cam-recorder. The system can provide to the user a quick visualization of the sphere, while the precise representation is computed off-line.

BMVC 2002 doi:10.5244/C.16.60

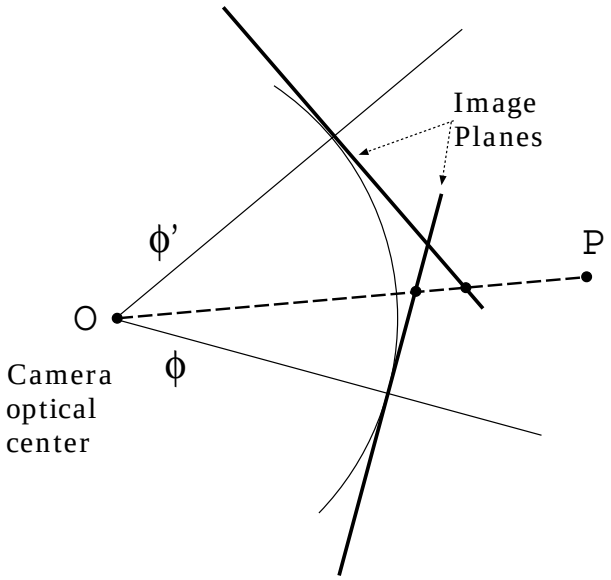

Figure 1: Geometry of pure camera rotations

In order to obtain a real time representation of this "painted" sphere, a real time registration algorithm has to be developed. This algorithm should provides 3D relative rotations of camera image planes to the mosaicing sphere. The optical center of the camera is supposed to be always at the same position (almost fixed).

Under the previous hypothesis and considering that the tracking algorithm gives the position of the camera for each new image, each pixel of any image can be characterized by its spherical coordinates and back projected on the sphere centered on the optical center of the camera (this fact is illustrated on Figure 1). Thus, any 3D point of the scene, situated at an unknown distance of the camera optical center, can be characterized by an optical ray (invariant by camera rotation). The spherical coordinates of this optical ray permit to back project the 3D point on the sphere.

Furthermore, we assume that the camera image plane does not rotate around its optical axis during the sequence. We also assume that the intrinsic parameters of the camera are known and fixed. This point is a matter of convenience since it reduces the number of degrees of freedom of camera displacements to 2. It also simplifies the equations of the bijection that maps a point from the sphere S into the image plane. In that case, the position of the image can be given using spherical coordinates. Therefore, the registration algorithm consists in providing, in real-time, the coordinates of the camera in the spherical coordinate reference of the sphere.

This article is made of 6 sections. Section 2 is devoted to a short presentation of the registration algorithm. A multi-tracking approach for 360 degrees wide registration is presented in section 3. The principles of image mosaicing algorithm are exposed in section 4. Finally, in section 5, some experimental results obtained from real video sequences are given.

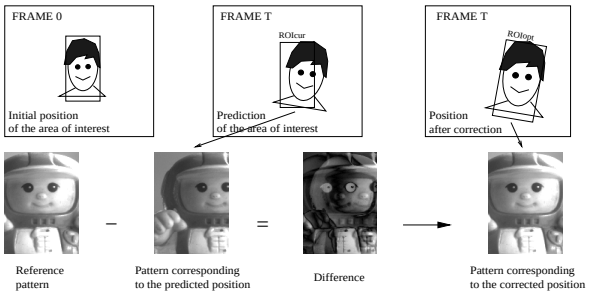

Figure 2: Principle of the difference image based approach.

2 Registration algorithm

2.1 Introduction

We need an algorithm able to register in "video real time" every images in the same 3D reference with a sub-pixel accuracy (to obtain an accurate mosaic).

Let us first describes several notations. The 3D world reference is denoted $R_w = (O_w, \vec{u}_\theta, \vec{u}_\phi, \vec{u}_\rho)$ and the current position of the camera (in the n^{th} image of the sequence) will only be characterized by its spherical coordinates (θ^n, ϕ^n), as the optical center coincides with O_w and as no rotation around the optical center is allowed. These two angles (respectively pan and tilt angles) give the position of the camera optical axis corresponding to the frame n. $(\theta^n, \phi^n) = (0, 0)$ when the camera reference is aligned with the reference of the camera, at the first frame of the sequence.

The goal of the registration algorithm is to provide estimations of (θ^n, ϕ^n) values for each new frame, in real time.

2.2 Principles

The proposed registration algorithm is based on our previous work on template matching described in [2]. The main idea of this algorithm is to relate camera displacements with the brightness variations in a region of interest of the image. In this paper we propose a new version of this approach allowing to track pan and tilt rotations of the camera.

The tracking problem is posed as the problem of finding the best (in least squares sense) set of parameter values describing the target motion $\delta\mu$ through a video sequence. Parameter variations $\delta\mu$ are written as a linear function of a *difference image vector* (this vector is obtained by sampling the target template and the template included in the predicted region of interest), as illustrated Figure 2. This approach is very efficient as motion can be directly deduced from the knowledge of the difference image vector.

The *difference image vector* is not computed on the whole region of interest but only at a limited number of points *called reference points*. The reference points are chosen in the area of the target template as the points having the highest norm of the gradient. Their coordinates are expressed by using spherical coordinates. In fact, the reference point corresponding to the image pixel of coordinates (u_i, v_i) will be characterized by the spherical coordinates (θ_i, ϕ_i) of the optical ray passing through the camera optical

center and image pixel. As the *reference pixels* move during the sequence, their spherical coordinates also change. For this reason, we will note (θ_i^n, ϕ_i^n) the spherical coordinates of the i^{th} *reference pixel* in the n^{th} image.

Giving the N *reference pixels* $\{p_i\}_{i=1..N}$, let $\vec{I}_{ref} = (I_{ref}(\theta_1^{ref}, \phi_1^{ref}), \ldots, I_{ref}(\theta_N^{ref}, \phi_N^{ref}))$ be the *reference vector*, where $I_{ref}(\theta_i^{ref}, \phi_i^{ref})$ represents the gray level value of the point characterized by the spherical coordinates $(\theta_i^{ref}, \phi_i^{ref})$ in the image where the target template is selected. Let $\vec{I}_n = (I_n(\theta_1^n, \phi_1^n), \ldots, I_n(\theta_N^n, \phi_N^n))$ be the corresponding vector obtained by sampling the n^{th} image of the sequence.

Variation of pixel intensities measured between the *reference* image and the n^{th} image are due to the rotations of the camera.

Let $E_n = (\theta^n, \phi^n)$ be the orientation of the current camera optical axis in R_w(at frame n). Given our hypotheses (camera optical center fixed and no rotation of the camera around its optical axis), these two parameters are sufficient to determine the camera displacements and to register images.

Thus, we would like to establish a relation between the variations of intensities of the reference points and the camera rotations. The camera rotations between frame $n-1$ and n are $\vec{\delta}E_n = \vec{E}_n - \vec{E}_{n-1}$. We can also write:

$$\vec{\delta}E_n = (\delta\theta, \delta\phi) \tag{1}$$

Let $\vec{\delta}I_n = \vec{I}_n - \vec{I}_{ref}$ be the difference between the intensities measured at reference points in frame n and those measured in the frame in witch the target template has been selected. $\vec{I}_{ref}$ corresponds to the reference template in the initial position of the camera. $\vec{I}_n$ corresponds to the intensities of the reference points measured in their last known position $(\theta^{n-1}, \phi^{n-1})$ in the frame n.

We are looking for the function f such that:

$$\vec{E}_n = \vec{E}_{n-1} + \overrightarrow{\delta E}_n = \vec{E}_{n-1} + f(\vec{\delta}I_n),$$

to compute iteratively the position of the camera for each new frame n.

A very straightforward and efficient computation of $\vec{E}_n$ can be obtained by assuming that f can be approximated by a linear transform [2]:

$$\vec{E}_n = \vec{E}_{n-1} + A\vec{\delta}I_n \tag{2}$$

where A is a matrix which results from the linearization of function f.

The spherical coordinates of the *reference pixels* are updated as follows:

$$\begin{aligned} \theta_i^n &= \theta_i^{n-1} + \delta\theta \\ \phi_i^n &= \phi_i^{n-1} + \delta\phi \end{aligned}$$

2.3 Learning stage

Equation $\vec{\delta}E_n = A\vec{\delta}I_n$ can be rewritten:

$$\begin{aligned} 0 &= (a_{11}, \cdots, a_{1N}, -1)(\delta_{I_1}, \cdots, \delta_{I_N}, \delta_\theta)^T \\ 0 &= (a_{21}, \cdots, a_{2N}, -1)(\delta_{I_1}, \cdots, \delta_{I_N}, \delta_\phi)^T \end{aligned}$$

where $a_{i1}, \cdots, a_{ij}, \cdots, a_{iN}$ $i \in [1,2]$ are the coefficients of 2 hyper-planes that can be estimated using a standard least square estimation.

To learn the matrix A, let us suppose that the current position of the camera is characterized by the vector $\vec{E}$. If this vector is disturbed in such a way that $\vec{E}' = \vec{E} + \vec{\delta}E$, each reference point (θ_i, ϕ_i) is moved to $(\theta_i + \delta\theta, \phi_i + \delta\phi)$ and the vector $\overrightarrow{I} = \{I(\theta_i + \delta\theta, \phi_i + \delta\phi)\}$ $i \in [1, N]$ is obtained by sampling the image in the region of interest. Then, the difference image $\vec{\delta}I = \vec{I} - \vec{I}_{ref}$ can be computed. This "disturbance" procedure is repeated N_p times, with $N_p > N$.

In this way, we collect N_p couples $(\vec{\delta}I^k, \vec{\delta}E^k)$. It is then possible to learn a matrix A such that $\sum_{k=0}^{k=N_p} (\delta E^k - A\delta I^k)^2$ is minimal.

The time required to compute A is a function of the cube of the number of reference pixels. This is why only a few number of reference pixels are used to compute $\overrightarrow{\delta I}$.

2.4 Tracking stage

As the axis of our 3D world reference R_w is aligned with the initial position of the optical axis of the camera, the first value of the camera location in R_w is given by $\vec{E}_0 = (0, 0)$.

The tracking stage of the algorithm consists in computing small camera rotations between images. When processing the frame n, the last known position of the camera is $\vec{E}_{n-1}$ and we want to compute $\vec{E}_n$. The matrix A and the brightness reference vector $\vec{I}_{ref}$ are supposed to be known (computed during the learning stage). To compute $\vec{E}_n$, we use equation (2): $\vec{E}_n = \vec{E}_{n-1} + A(\vec{I}_n - \vec{I}_{ref})$.

The registration procedure is therefore reduced to two straightforward operation :

- sampling reference points (vector $\vec{I}_n$)
- computation of a matrix-vector product.

2.5 Correction of lens distortions: from image coordinates to spherical coordinates

In the preceding section, the problem of finding the geometric displacement between two images is addressed, but without taking into account the lens distortions. As we pretend to obtain sub-pixel accuracy in the displacement estimate, it is necessary to take into account the geometrical distortions of the camera.

The distortions are estimated using the calibration procedure described in [3]. This procedure computes the intrinsic parameters of the camera: f (focal length), (u_0, v_0) (image principal point), (d_x, d_y) (pixel size) and distortion parameters $(D_x(x,y), D_y(x,y))$ sum of radial part and a tangential part, from several images of a calibration pattern.

Knowing these various parameters, the spherical coordinates (θ, ϕ) of an image points (u, v) can be obtained from the following equations:

$$\begin{aligned} \theta &= \arctan\left(\sqrt{\frac{f^2}{x^2+y^2}}\right) \\ \phi &= \arctan\left(\frac{y}{x}\right) \end{aligned}$$

with:

$$\begin{aligned} x &= (u - u_0)d_x - Do_x \\ y &= (v - v_0)d_y - Do_y \end{aligned}$$

These corrections provide a better stability of the registration algorithm, and ensure a good accuracy.

3 Multiple tracking

Camera rotations change the positions of reference points in the image. Since we track these points by sampling their brightness, they have to be always visible. As we want to cover the whole sphere, several successive templates will be tracked in turn. Each template is a small square zone of the image, in which the reference points are taken. Therefore, the algorithm presented in the previous section will only be able to track camera displacements while this region of interest is visible.

It is why there is a need to use – in parallel – the tracking of several templates to ensure the continuity of the registration. The tracked templates, which cover the image, will be re-initialized when the templates will go outside of the image.

Each tracker provides its own estimate of the camera rotation for every frame of the sequence.

If we use N_t trackers, we obtain N_t estimations of the camera rotations in R_w. Given these Nt estimations $(\delta\theta_i, \delta\phi_i)$, we keep the median value of each parameters among the N_t computed values as the final estimation of the camera displacement.

When the region of interest associated to a given tracker goes out of the image, a new tracker is initialized while the other trackers keep working normally. This procedure ensures a continuous tracking.

3.1 Trackers management

The positions of the tracked templates have to be determined automatically. Furthermore, it should be possible to add or remove trackers during the registration of images.

At first, one can observe that it would be very inefficient to place two trackers on the same region of the image. Indeed, a better estimation of camera displacements is obtained when the various trackers are well dispatched all over the image. To ensure a good repartition, we define a grid on the image such that each bucket contains a tracker. When a tracker steps out of its bucket, it's replaced by an other one.

We maintain two lists of trackers: the first one is a list of *active trackers*, those which actually track the camera motions ; the second one is a list of *inactive* trackers. When a tracker goes out of its bucket, it is removed from the active tracker list and moved to the inactive tracker list.

The creation of a new tracker in a bucket that do not have any tracker is driven in two stages. First, a tracker whose template is belonging to this bucket is searched into the list of inactive trackers. If such a tracker exists it is associated with the bucket. If such a tracker does not exists, a new tracker is computed. In both cases, the tracker newly associated to the bucket is pushed into the list of active trackers. This method allows to reduce the number of initializations and speed up the algorithm.

When a new tracker is created, it is necessary to define its position in the bucket. A first solution would be to center the region of interest in the bucket. After some tests, we concluded that this solution was not optimal. Indeed, the precision of the template matching algorithm depends on the texture of the region of interest: the more textured the region is, the better the tracking is. It is why we use a local analysis of the bucket to determine how good a region of interest is. For this, we sample the bucket and compute a variance indicator for each region position and retain the region maximizing the indicator.

Using these various strategies, the proposed system is able to register images all the view sphere round, with a sub-pixel accuracy.

4 Image mosaicing

4.1 Creation of a spheric image

Thanks to the multiple tracking algorithm, we know the camera rotations (θ^n, ϕ^n) in R_w, at every frame n. Giving (θ^n, ϕ^n), we want to construct the part of the spheric image corresponding to the current frame.

Thus, we have to establish the relation between a point on the sphere of coordinates (θ, ϕ) and its corresponding point (u, v) in the n^{th} image.

This relation is given by the following equations:

$$\begin{aligned} u &= u_0 + \frac{f\cos(\phi+\phi^n)}{d_x\tan(\theta+\theta^n)} \\ v &= v_0 + \frac{f\sin(\phi+\phi^n)}{d_y\tan(\theta+\theta^n)} \end{aligned}$$

Given a point (θ, ϕ) of the spheric image, we can compute its projection on the n^t image. Nevertheless, such a projection method gives bad results if camera distortions are not taken into account.

As explained in section 2.5 to correct camera distortion we use the distortion parameters obtained by calibrating the camera. The distortion parameters have been taken into account to make the previous equation more accurate.

5 Results

The proposed approach have been tested on several video sequences. The results are view spheres that are difficult to represent in 2D. In this article we have decided to produce panoramic image, by applying pure panning motions.

The 2D panoramic image have been obtained by representing on a plane the computed sphere. We do not project it but only visualize it using spherical coordinates.

Figure 3 present two panoramic images that have been obtained by using the proposed algorithm. Some details of these images show the accuracy of the registration.

Other results in color as well as the original video sequences can be obtained on the web: http://wwwlasmea/Personnel/Frederic.Jurie/mosaics.html (without dot after www).

Figure 3: Two panoramic images and several details. Other results in color as well as the original video sequences can be obtained on the web: http://wwwlasmea/Personnel/Frederic.Jurie/mosaics.html (without dot after www).

6 Conclusion

In this article , we propose an original method for image mosaicing. The key idea is to use a real time template matching to track camera displacements, allowing to register each one of the image sequences into the view sphere. We give examples proving that our method is efficient and accurate. We plan to estimate the focal length in real time, in order to produce super-resolution images. We can also mention the fact that a better mesh for spheric image representation has to be used. Up to now we use a rectangular mesh which is not really adapted.

References

[1] R. Benosman and S.B. Kang. *Panoramic Vision: Sensors, Theory, Applications*. Springer, 2001.

[2] F. Jurie and M. Dhome. Real time template matching. In *Proc. IEEE International Conference on Computer vision*, pages 544–549, Vancouver, Canada, July 2001.

[3] J.M. Lavest, M. Viala, and M. Dhome. Do we really need an accurate calibration pattern to achieve a reliable camera calibration? In *ECCV98*, pages xx–yy, 1998.

[4] S.K. Nayar. Catadioptric omnidirectional cameras. In *CVPR97*, pages 482–488, 1997.

[5] H.Y. Shum and R. Szeliski. Panoramic image mosaics. In *Microsoft*, 1997.

[6] H.Y. Shum and R. Szeliski. Systems and experiment paper: Construction of panoramic image mosaics with global and local alignment. *IJCV*, 36(2):101–130, February 2000.

[7] Y. Xiong and K. Turkowski. Creating image based vr using a self-calibrating fisheye lens. In *CVPR97*, pages 237–243, 1997.