Fully Automated Texture Tracking Based on Natural Features Extraction and Template Matching

Rafael Bastos
ADETTI
Av. das Forças Armadas,
Edifício ISCTE 1600-082 Lisboa
(+351) 21 782 64 80, Portugal
rafael.bastos@sapo.pt

José Miguel Salles Dias
ADETTI
Av. das Forças Armadas,
Edifício ISCTE 1600-082 Lisboa
(+351) 21 782 64 80, Portugal
miguel.dias@iscte.pt

ABSTRACT

In this work we propose a novel approach to real-time texture tracking and registration, based on natural feature extraction from planar objects and template matching, Our method is oriented to planar objects with arbitrary textures but with rectangular topologies and well contrasted contours and does not require any external fiducial marker, either for the set-up or the tracking phases. Once the initial pose condition is obtained, previous planar object information is used to compute subsequent planar object's pose, so that the time coherence of the input video stream is exploited. Our system is completely automated and produces real-time efficient tracking which can be applied to entertainment AR applications and other. The paper discusses also the novelty of the approach, in relation to other existing texture tracking algorithms.

Categories and Subject Descriptors

I.5.4 [Pattern Recognition]: Applications -- Computer vision; I.4.8 [Image Processing and Computer vision]: Scene Analysis -- Motion, Tracking; J.7 [Computer in other systems]: consumer products.

General Terms

Algorithms, Measurement, Performance, Experimentation, Theory.

Keywords

Augmented Reality, Texture Tracking, Template Matching, Texture Reconstruction.

1. INTRODUCTION

The camera registration or tracking problem in Augmented Reality-AR tracking systems context has been a research topic of considerable interest and constant grow in the literature. There are a variety of different methods to accomplish this goal, with the first ones being introduced in 1992 by Caudel and Mitzell [1]. The proliferation of vision-based tracking techniques is due to the fact that they work well in real time and they only incur in one cost: the processor's cost. We propose a novel approach to texture tracking based on natural feature extraction without the need of any kind of extra information, like fiducial markers [2], to extract planar objects pose, even for the set-up phase [3]. Our technique combines methods such as template matching, homography computation, like in [4], texture reconstruction by back-projection (as in [3]) and Kalman filtering. Our system has an off-line-automated stage where all image good features to track are extracted, as well as all the ones needed to obtain the planar object initial pose. Then the algorithmic process can be divided in two phases: the set-up phase, which obtains the initial camera pose and the tracking phase. In the tracking phase two methods are adaptively applied: one, based in simple contour motion detection, that requires that the complete planar object must be visible, and the other one, based on texture reconstruction and template matching, which only requires the partial visibility of the planar object. The system is completely automated and produces real-time efficient tracking.

2. SYSTEM ARCHITECTURE

Three main modules comprise our system: Video Capture Module, Texture System Manager and Open GL Rendering Module. The Video Capture module is implemented with Microsoft DirectShow filters, working with most Web Cameras or even with DV Firewire Camcorders. The Capture module works with 320x240 video resolution and can achieve 40 frames per second in video acquisition rate. This Video Capture module delivers each video frame to the Texture System Manager, which is responsible for all image-processing routines. These routines can be structured into three phases: Texture Automation Processing phase, Set-Up phase and Tracking phase. The Texture System Manager is also responsible for handling a list of textures to track, so that multi-texture tracking can be supported. Upon successful feature tracking (with a minimum of five tracked features), the Texture System Manager only has to deliver the texture pose to the Open GL Rendering Module, in order to register any 3D virtual information or object at the specified texture.

3. Algorithm Description

Our texture tracking algorithm includes a Texture Processing Phase (an off-line natural feature extraction technique) and a subsequent Real-Time Feature Tracking method, which is

subdivided in a Set-Up Phase that computes the initial planar object pose and a Feature Tracking Phase, which tracks the planar object pose in time.

3.1 Texture Processing (Off-Line) Phase

The Texture Processing Phase is the first stage of the technique and is performed offline. At this stage, we acquire the planar object image and process this global template texture image in order to collect all the good features to track. In order for a feature to be considered valid, we must guarantee that its template pattern can't be similar to any other feature's template pattern. So, we can guarantee that these feature are unique in global template subspace. Another requirement is that features must have a balanced distribution over the image plane to ensure robust pose retrieval. The first step of the offline stage is feature extraction from the template image. For this purpose we compute the minimum eigen values of the image, which give us the potential localization of the good features to track. After that, for each extracted feature, a template match is done against all the others. If any template matching score is higher than 0.9 (90%), then this feature is discarded. If all scores are smaller than 0.9, then we proceed to the next step. We must now ensure that this unique feature is well distributed, or in other words, that there aren't more features in a 10 pixels radius. If this fact reveals to be true, then we have just found a feature that is unique in the whole image, and that can ensure robust pose retrieval. At the end of this phase, a data file is written containing all features positions, so that these can be used by the main system. An external program developed for this purpose, called Texture Automizer, is responsible for the whole off-line process.

3.2 Set-Up Phase

The Set-Up phase is dedicated in finding the initial texture (planar object) pose, so that the subsequent Feature Tracking Phase may begin.

3.2.1 Contour Extraction and Initial Pose Computation

We start by extracting all line contours in the image using an adaptive threshold technique [5] (see Figure 1). This technique computes and updates the local pixel threshold along the raster-scanned line instead of a global image threshold computation, in order to solve the problems that occur in image binarization, caused by illumination, contrast, noise and other image degradation sources. Then these line contours are traced by following polylines with a complete 3x3-structuring element and by keeping the ones formed by four points. Small area contours are rejected, so that the main focus of the subsequent processing is the texture to track. The resulting contours are labeled, and the inner contours are discarded. This process is quite straightforward. By analyzing each traced contour, we can arrange them by the proximity of each centroid. With this scheme, we can find the contour that has the largest area lying around a certain centroid. By default, if no noise is induced, we will find two square contours for each texture, and only the outer one is considered valid. For each valid contour, its pose is computed using a classical pinhole camera model and the Five-Point Pose Algorithm with homography computation to determine the texture's pose (camera extrinsic parameters), which is described in [6]. This algorithm requires the knowledge of at least five 3D world/2D projected image point correspondences, and of the camera intrinsic parameters. For that purpose we use the four corners of the rectangular contour and its centroid. After evaluating the rectangle pose, in relation to the camera reference frame, we still have an indetermination in the orientation of such basic shape, due to its symmetry. In Figure 3 we depict the problem: in the right hand side, we see the four possible orientations of the rectangular image, while just one is valid (the left hand side image). To solve this problem, given the rectangular planar object pose, we must compute its image back-projection and then perform template matching with each of the four images of the right hand side of Figure 3, to properly classify the orientation of the planar image in question. In the following sub-sections, we discuss such techniques.

3.2.2 Back-Projection and Template Matching

By knowing the camera intrinsic and extrinsic parameters of the previous pose, which relates the world coordinates of the rectangular image with the pixel coordinates of that projected rectangle, we are able to evaluate the current back projected image pixels, with the pixels obtained in the current video stream frame, at the computed projected coordinates. This back-projected image is tested for template matching, with four pre-computed global template images oriented, respectively, by 0º, 90º, 180º and 270º (Figure 3), to determine the correct orientation of the rectangular texture detected in the current video stream. In general terms, the Template Matching algorithm is used to determine the level of similarity between two images and there are various possible methods to accomplish this. We have chosen a method [7], which computes the correlation coefficient ρ. This noise and luminance invariant coefficient is the measure of dissimilarity between the template image P and the image I, acquired from a live video stream, and was adopted in both phases of our Texture Tracking algorithm. When used in the second phase (Feature Tracking), we propose the use of circular patches, with an area of 16x16 pixels, instead of square ones for the Template Matching algorithm, in order to maximize the degree of correlation between patches, since feature points are centered spatially in regions of high frequency. These circular patches revealed more tolerance to small degrees of rotation and scaling. A threshold value of 0.9 (90%) was chosen. After the pose retrieval is achieved, two Kalman filters are initialized with the translational and rotational degrees of freedom of the computed texture pose, so that a more stable assumption about extrinsic camera parameters can be made in the tracking phase. Our implementation of Kalman filtering uses Open Computer Vision Library routines and data structures [8].

Figure 3 – Back-projected image and selected orientation.

3.2.3 Pose Retrieval

By knowing which orientation has the higher score of Template Matching (Figure 3), the camera pose of the current frame can be computed using the same approach as referenced in section 3.2.1 (the classical pinhole camera model and the Five-Point Pose Algorithm with homography computation), when we have evaluated the initial pose.

3.3 Feature Tracking Phase

At the start of the Feature Tracking phase, it's assumed we have the previous texture pose information. Now the system's

processing path is divided in two distinct branches: when contours are visible and when contours are occluded. For each frame at this stage, contours are extracted like described in section 3.2.1. If any four contour points is found, then we follow the first branch of the tracking path: *tracking with contours*, if not, we proceed to the *tracking without contours* algorithm.

3.3.1 Tracking with contours

When *tracking with contours*, we must verify if the found contour corresponds to the previous one identified in the Set-Up phase or in the Tracking phase (with contours), subject to a rigid body motion. For that, we re-project the known texture's bounding rectangle vertexes (of the previous pose) into the 2D image coordinate space using the previous pose camera information. Then, a simple comparison is made in pixel coordinates between the re-projected points and the current contour vertexes. If this computation gives a result inferior to 20 pixels for each vertex, then we can assume that the compared contour corresponds to the texture's contour. On the other hand, if this comparison result is not satisfactory, then we proceed to the *tracking without contours* technique, which uses feature information (section 3.3.2). When contours are matched, we need a 5th point in order to compute the texture's homography and derive the texture pose, using the technique described in section 3.2.3. Instead of using the contour centroid, which will induce a tracking error, we use a more precise point that can be found by matching a known feature with the one closest to the texture center. For that, we use the technique explained in section 3.3.2, that first reconstructs the known feature patch using the previous camera pose and then performs template matching in an area close to the current texture centre, to find the feature that occupies that centre.

Figure 4 – Reconstructed feature patches (top-left); Matched features in the live image (top-right); Teapot registration based on computed pose (bottom).

3.3.2 Tracking without contours

If contour tracking fails, another algorithm based on feature information is used. For that purpose, artificial texture reconstruction based on the previous camera pose is preformed, in order to simulate an approach to the current captured frame. After this, we proceed to a process of selection of "good features to match" based on a heuristic approach. Upon selection of such "good features to match", template matching is done between the selected features of the reconstructed texture and features within an area of 26x26 pixels, of the current video frame. If all five features match succeed, then a new texture pose is retrieved. If less then five features are matched, then we go back to the beginning of the Set-Up Phase, which requires that the full texture be visible.

3.3.3 Texture Reconstruction

In order to perform a closest match on subsequent frame features, we use the previous camera extrinsic parameters to reconstruct the current texture pose. As mentioned, template matching is done between each feature located in the live video image and each feature located in the reconstructed texture image. Feature's matching score will be stronger because scale and rotation factors won't have a large impact from one frame to another, when working in real-time. By projecting world texture coordinates into image (pixel) coordinates, we can compute an approach to the current texture pose. Based on this reconstructed texture approach and by knowing were all features identified from the template image, should lie in the current frame, we can now start a dynamic and adaptive template matching. This time feature selection will follow certain rules, so that results can be more stable and with less induced error.

3.3.4 Feature Selection and Matching

With the purpose of updating the system with a fresh texture pose, a template matching process and a correspondent feature selection must be done. There are certain steps when choosing which features are "good to match" and will be used in this process:

a) Features must lie within the video frame;
b) The first feature must be the one closest to the center of the image;
c) The second feature must be the one closest to the top left of the image;
d) The third feature must be the one closest to the top right of the image:
e) The fourth feature must be the one closest to the bottom left of the image:
f) The fifth feature must be the one closest to the bottom right of the image.

From step b) to step f), when selecting features, template matching is performed between each feature located in the live video image and each feature located in the reconstructed texture image. Features on the reconstructed image are selected with a circular area of 16x16 pixels (see 3.2.2, Template Matching), and searched in the current live video image within an area of 26x26 pixels, centered at the predicted feature location. With the concern of optimizing the algorithm and reducing the processing cost, minimum eigen values are computed within this region (26x26) of interest in the live image and we assume that potential features will reside in the location of these eigen values. After this identification, template matching is done only with patches centered in these potential features. If a given feature has a matching score less then 0.9 (90%), then it is considered to be lost

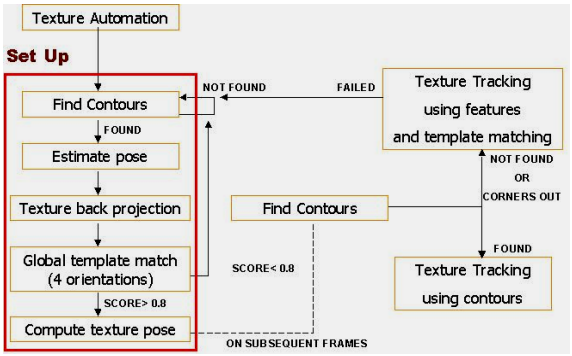

Figure 5 – Flow diagram of the Fully Automated Texture Tracking Algorithm.

Processor	Pentium IV / 2.66 GHz
RAM	457.712 KB RAM
Graphic Adapter	ATI Mobility Radeon IGP 340M AGP, 64 MB
Hard Disk	60 GB
OS	Window 2000 Service Pack 4

Table 1 – Used hardware.

in the current frame and the search continues in the same step, until a successful match is found, or the time step expires with no success found. If less then five features matches are achieved, then we assume that feature tracking fails and we proceed to a new Set-Up Phase. If five features are matched, we proceed to the pose retrieval process, and feed the rotation and translation Kalman filters with the new values taken from the camera extrinsic parameters of the pose. At this stage, by achieving a positive result in tracking, we may register any kind of 3D object using the texture pose information (Figure 4).

4. PRECISION AND PERFORMANCE TESTS

Table 1 shows the hardware configuration used when performing precision and performance tests.

4.1 Precision Tests

Precision tests of our texture-tracking algorithm were based on the simulation of the pose a 3D plane object (with a texture), subject to translation and rotation degrees of freedom, much like the ones that occur when using the system with a HMD. For this simulation we have used the image texture reconstruction technique mentioned in this paper. To achieve this, we have recorded the live captured video output (in digital format) taken from the camera. After this, we have stored all the camera extrinsic matrixes (one per each frame) computed by our algorithm in a data file (in off-line mode). Each one of these matrixes was used to reconstruct synthetically a texture, and the full set generated a synthetic video stream. This stream was then used to feed our texture tracking algorithm, so that the obtained camera poses could be mathematically compared with the known simulated poses, of the texture plane. For this simulation test, 3000 frames were taken into account, and for each degree of freedom all computed and simulated rotation and translation values were stored. All computed values suffered Kalman filtering, so that tracking errors could be minimized. After that stage, they were analyzed and the error was computed. As we can see from the Table 1 results, translation has an overall average error of 1.13 millimeters that is visually unnoticed in real-time applications. Rotation has an overall average error of 1.45 degrees that can be sometimes noticed in real-time applications, when the user performs fast manipulations of the planar texture object.

Average Error	**Over X (Err_x)**	**Over Y (Err_y)**	**Over Z (Err_z)**	**Overall ($OvErr$)**
Translation (mm)	**0.58**	**0.57**	**2.23**	**1.13**
Rotation (deg)	**1.28º**	**1.50º**	**0.66º**	**1.45º**

Table 2 – System Pose Error when using feature/texture information

Rendering velocity	**54 fps**
Total Processing cost	18 ms
- Contour Extraction	8 ms
- Back-Projection	5 ms
- Template Matching	4 ms
- Pose Estimation	1 ms

Table 3 – Set-Up Phase Performance Results.

Rendering velocity	**82 fps**
Total Processing cost	12 ms
- Contour Extraction	8 ms
- Feature Reconstruction	2 ms
- Template Matching	1 ms
- Pose Estimation	1 ms

Table 4 –Tracking with contours performance.

4.2 Performance Tests

Performance tests were made using template texture images with 300x300 dpi and a resolution of 640x480 pixels. The input video resolution was 320x240 and the output video resolution was 1024x768, in full screen mode. The graphics library used for rendering was Open GL. As we can see from the results shown in Tables 3-5, the overall algorithm works in real time in all phases, achieving 82 fps in the Tracking Phase, when contours are visible. Our system is prepared to work with real time frame capture and the limiting factor in the obtained tracking, is thus the camera acquisition frame rate.

Rendering velocity	**62 fps**
Total Processing cost	16 ms
- 5 Point Feature Reconstruction	10 ms
- Template Matching	5 ms
- Pose Estimation	1 ms

Table 5 –Tracking without contours performance.

5. CONCLUSIONS

We have proposed a novel approach to texture tracking based on natural feature extraction in unprepared scenes without the need of any kind of extra information previously added to the scene, like fiducial markers to extract the planar textured objects pose, including for the set-up phase [3]. Our method has been shown to work in real-time and in indoor and outdoor environments. However, it is restricted to planar surfaces, which may be a problematic constraint in complex environments. The system produces real-time efficient and robust tracking. Recent state-of-the-art advances [9], have adopted a model-based method to full 3D object tracking (in contrast to our method which is just oriented to planar objects), where rough CAD model of parts of the real scene are known beforehand. We intend also to generalize our technique to 3D model-based tracking of indoor and outdoor scenes.

6. REFERENCES

[1] Caudell, T.P. and Mizell, D.W., "Augmented Reality: An Application of Head-Up Display Technology to Manual Manufacturing Processes," in Proceedings IEEE Hawaii International Conference on Systems Sciences, 1992, Kauai, HI, IEEE, 0073-1129-1/92, pp. 659-669. January 1992.

[2] http://www.hitl.washington.edu/artoolkit/

[3] Kato, Hirokazu , Tachibana, K., Billinghurst, M., Grafe, M., "A Registration Method based on Texture Tracking using ARToolKit", in The Second IEEE International Augmented Reality Toolkit Workshop, Nishi-Waseda Campus, Waseda University, Tokyo, Japan, 7th October 2003.

[4] G. Simon, A. Fitzgibbon and A. Zisserman, "Markerless Tracking using Planar Structures in the Scene", in ISMAR' 00 Conference Proceeding, 2000.

[5] R. Sittisak, R. Yuttapong, "Adaptative Thresholding of Document Images Based on Laplacian Sign", International Conference on Information Technology: Coding and Computing, 2001.

[6] D. Nistér. "An Efficient Solution to the Five-Point Relative Pose Problem", IEEE Conference on Computer Vision and Pattern Recognition, Volume 2, pp. 195-202, 2003.

[7] Charles B. Owen, Fan Xiao, and Paul Middlin, "What is the best fiducial", in The First IEEE International Augmented Reality Toolkit Workshop, September 29, Darmstadt, Germany, 2002.

[8] http://sourceforge.net/projects/opencvlibrary/

[9] Comport, A. I., Marchand, E., Chaumette, F., A realtime tracker for markerless augmented reality, IRISA - INRIA Rennes, France, in The Third IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR' 03), Tokyo, Japan, Oct. 7 - Oct. 10, 2003.