On Computer Vision for Augmented Reality

Conference Paper · August 2008

DOI: 10.1109/ISUVR.2008.10 · Source: IEEE Xplore

CITATIONS
12

READS
506

1 author:

Vincent Lepetit
Université Bordeaux 1
243 PUBLICATIONS **15,902** CITATIONS

SEE PROFILE

On Computer Vision for Augmented Reality

Vincent Lepetit
École Polytechnique Fédérale de Lausanne (EPFL)
Computer Vision Laboratory
CH-1015 Lausanne, Switzerland
vincent.lepetit@epfl.ch

Abstract

We review some recent techniques for 3D tracking and occlusion handling for Computer Vision-based Augmented Reality. We discuss what their limits for real applications are, and why Object Recognition techniques are certainly the key to further improvements.

1. Introduction

Computer Vision has great potential for Augmented Reality applications. Because it can rely on visual features that are naturally present to register the camera, it does not require engineering the environment and is not limited to a small volume, like magnetic, mechanical or ultrasonic sensors are. Moreover, it is certainly not conceited that only Computer Vision can assure an alignment between the real world and the virtual one with an accuracy of the order of the pixel, since it is precisely on this information it relies on.

And still, not real AR applications based on exist. Many applications have been foreseen for many years now—in medical visualization, maintenance and repair, navigation aid, entertainment—and yet markers-based applications are the only successful ones. However markers are a limited solution because they still require engineering the environment, work on a limited range, and end-users often do not like them.

The reason of such absence is quite obvious. Most of the current approaches to 3D tracking are based on what can be called *recursive tracking*. Because they exploit a strong prior on the camera pose computed from the last frame, they are simply not suitable for practical applications: First, the system must either be initialized by hand or require the camera to be very close to a specified position. Second, it makes the system very fragile. If something goes wrong between two consecutive frames, for example due to a complete occlusion of the target object or a very fast motion, the system can be lost and must be re-initialized in the same fashion.

Sensor fusion with GPS and magnetic sensors are also very promising and impressive results have been demonstrated [5]. However, such approaches are limited to outdoor applications and are not adapted to augmentation of mobile objects.

Recently, several works relying only on Computer Vision but able to register the camera without any prior on the pose have been introduced. An example is depicted Fig. 1. They are not only suitable for automated initialization, they are fast enough to process each frame in real-time, making the tracking process extremely more robust, preventing loss of track and drift. We shall call the approach *tracking-by-detection*.

Robustness of camera registration is not the only aspect of Augmented Reality where object recognition techniques can contribute. Handling occlusions between the real objects and the virtual ones is another one. Currently there is no satisfactory real-time methods. When it comes to occlusion masks, errors of only a few pixels are easily noticeable by the user and intensity or 3D reconstruction-based approaches can only produce limited results. It is certainly only with a high-level interpretation of the scene that the problem can properly be solved.

In the remainder of the paper, we review some recent techniques for camera registration and occlusion handling, discuss their limitations, and try to give directions of research.

2. Tracking-by-Detection

In tracking-by-detection approaches, feature points are first extracted from incoming frames at run-time and matched against a database of feature points for which the 3D locations are known. A 3D pose can then be estimated from such correspondences, for example using RANSAC to eliminate spurious correspondences.

It should be noted that it does not imply that recursive tracking methods become useless. Tracking-by-detection

Figure 1. The advantages of tracking-by-detection approaches for Augmented Reality. The target object, the book in this example, is detected in every frame independently. No initialization is required from the user, and tracking is robust to fast motion and complete occlusion. The objects 3D pose can be estimated and the object can be augmented.

tends to be less accurate while recursive approaches usually have a narrow but peak basin of convergence that makes them more accurate. One strategy can then be to first estimate the pose with a detection method, and then refine it with a more traditional approach. Exploiting temporal consistency is also still interesting but not straightforward to do if one does not want to re-introduce drift.

The difficulty in implementing tracking-by-detection approaches comes from the fact that the database images and the input frames may have been acquired from very different viewpoints. The so-called *wide baseline* matching problem becomes a critical issue that must be addressed. One way to establish the correspondences is to use SIFT, as it was done in [6]. Another approach that proved to be very fast is to use a classifier to recognize the features.

In the following, we describe such an approach, and an extension that relaxes somehow the need for texture. We also discuss the limits.

2.1. A Simple Classifier for Keypoint Recognition

Several classification methods have been proposed for keypoint recognition. We quickly describe here the method of [3] because of its simplicity and efficiency.

A database of H prominent feature points lying on the object model is first constructed. To each feature point corresponds a class, made of all the possible appearances of the image patch surrounding the feature point. Therefore, given the patch surrounding a feature point detected in an image, the task is to assign it to the most likely class. Let $c_i, i = 1, \ldots, H$ be the set of classes and let $f_j, j = 1, \ldots, N$ be the set of binary features that will be calculated over the patch. Under basic assumptions, the problem reduces to finding

$$\hat{c_i} = \underset{c_i}{\operatorname{argmax}} P(f_1, f_2, \ldots, f_N \mid C = c_i) \,, \qquad (1)$$

where C is a random variable that represents the class. In [3], the value of each binary feature f_j only depends on the intensities of two pixel locations $\mathbf{d}_{j,1}$ and $\mathbf{d}_{j,2}$ of the image patch:

$$f_j = \begin{cases} 1 \text{ if } I(\mathbf{d}_{j,1}) < I(\mathbf{d}_{j,2}) \\ 0 \text{ otherwise} \end{cases}$$

where I represents the image patch. Note that the values of these features are unchanged when an increasing function is applied to the intensities of the image patch. That makes the final method very robust to light changes.

But since these features are very simple, many of them are required for accurate classification ($N \approx 300$), and therefore a complete representation of the joint probability in Eq. (1) is not feasible. The Ferns approach of [3] partitions the features into several groups, and the conditional probability becomes

$$P(f_1, f_2, \ldots, f_N \mid C = c_i) = \prod_{k=1}^{M} P(F_k \mid C = c_i) \,. \quad (2)$$

In practice, it appears that the locations $\mathbf{d}_{j,1}$ and $\mathbf{d}_{j,2}$ of the features can be picked at random, making training particularly simple. The terms $P(F_k \mid C = c_i)$ are estimated by computing the features on training samples of each class. From a small number of images, many new views can be synthesized using simple rendering techniques as affine deformations, and extract training patches for each class. White noise is also added for more realism.

The resulting method is extremely fast, and very simple to implement. Rotation and perspective invariance are directly learned by the classifier, and no parameter really needs tuning.

2.2. Relaxing the Need for Texture

The method described above produces a set of 3D-2D correspondences from which the object pose can be computed. In theory, when the camera internal camera are known, only three—or four to remove some ambiguities—correspondences are needed. In practice, much more are required to obtain an accurate pose and to be robust to erroneous correspondences. That implies that the previous approach is limited to relatively well-textured objects in practice.

However it should be noted that the previous approach only aims to establish point-to-point correspondences, while the appearance of feature points, not only their locations, also provide a cue on the orientation of the target object. As shown in Fig. 2, [1] recently introduced

Figure 2. Relaxing the need for texture. [1] not only matches feature points, but also estimates their local pose. These local poses can be extended to retrieve the object pose. As a result, a single feature is often enough to make the method very robust to occlusion (right image), and suitable for low textured objects.

a method to efficiently estimate the local transformations around feature points and exploit them to compute the object pose. Actually, a single feature point becomes enough to estimate this pose, relaxing the need for textured objects.

The method described in [1] performs in three steps. First, a classifier similar to the one described in Section 2.1 provides for every feature point not only its class, but also a first estimate of its transformation. This estimate allows carrying out, in the second step, an accurate perspective rectification using linear predictors. The last step checks the results and remove most of the outliers.

The transformation of the patches centered on the feature points are modeled by a homography defined with respect to a reference frame. The first step gives an initial homography estimate $\widehat{\mathbf{H}}$ of the true homography $\overline{\mathbf{H}}$, and the hyperplane approximation of [2] is used to efficiently estimate the parameters $\widetilde{\mathbf{x}}$ of a corrective homography:

$$\widetilde{\mathbf{x}} = \mathbf{A}\left(\mathbf{p}(\widehat{\mathbf{H}}) - \mathbf{p}^*\right) , \qquad (3)$$

where

- $\mathbf{A}$ is the matrix of the linear predictor, and depends on the retrieved class c_i for the patch. It can be learned from a training set.
- $\mathbf{p}(\widehat{\mathbf{H}})$ is a vector that contains the intensities of the original patch $\mathbf{p}$ warped by the current estimate $\widehat{\mathbf{H}}$ of the transformation.
- $\mathbf{p}^*$ is a vector that contains the intensity values of the patch under a reference pose.

This equation gives the parameters $\widetilde{\mathbf{x}}$ of the incremental homography that updates $\widehat{\mathbf{H}}$ to produce a better estimate of true homography $\overline{\mathbf{H}}$.

2.3. It Is Not Enough

Many challenges remain. To see that, let's say one want to detect the car in Fig. 3, and estimate its 3D pose, for example to add a virtual logo on it. While humans have no problem seeing the car, no existing method is able to do it. The surface of the car is shiny and smooth, and most of the feature points come from reflections and are not relevant for pose estimation. The only stable feature points one can hope to extract (on the corners of the windows, on the wheels, on the lights...) will be extremely difficult to match because of complex light effects and the transparent parts. While this example is a bit extreme, everyday objects are indeed often specular, not well textured and of complex shape. The recent success should not make researchers in AR overlook the difficulties that remain to be solved.

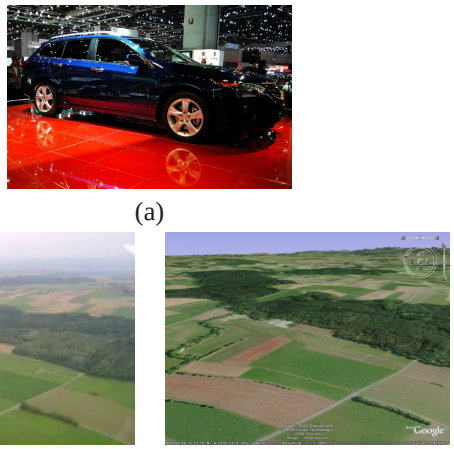
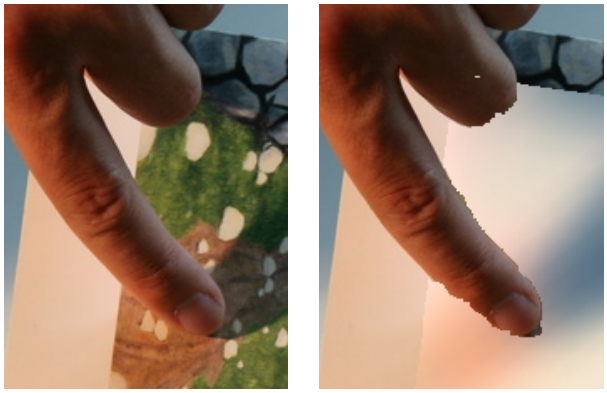


Figure 3. Some challenges for Computer Vision. (a) An example of a difficult object for tracking-by-detection approaches. The reflections, the absence of texture, and the smooth edges make the car difficult to detect. (b) A real image and (c) an image rendered by GoogleEarth of the corresponding scene. No existing method is able to establish correct correspondences between the two images.

Let's now consider an application for navigation aid on a large scale. Many such applications have been imagined by the Augmented Reality community. Sensors such as GPS and magnetic sensors can of course be of great help, but vision is still needed for accuracy. For such an application to actually work, the system must be robust not only to perspective changes, but also to complex light changes with the time of day and the time of year, the change of appearance of the vegetation between summer and winter, and so on.

To illustrate the problem, we compare in Fig. 3(b) and (c) a real outdoor image with an image rendered using the textured model of the same scene of GoogleEarth. Once again, it is not particularly difficult for a human to realize it is the same scene seen from the same viewpoint. However, no existing method is able to establish correct correspondences between the two images. Between the two images, the sun position, the field natures,... changed and that makes texture based methods fail. For real applications, a vision-based tracking system should be able to be robust to such changes.



(a) (b)

Figure 4. A real image and the corresponding of visibility and lighting maps computed by [4]. The method is robust to complex light changes, but the small errors along the finger boundaries break the illusion.

3. Handling Occlusions

Another problem that remains to be solved is the correct handling of occlusions between real and virtual objects. When a 3D model of the real occluding objects is available, it is relatively easy to solve. However, it is not possible to model the whole environment, in particular when the occluding objects can be pedestrians or the user hands. Very few methods have been proposed yet. We describe below an existing method and its limits.

3.1. A Subtraction Method

[4] starts by registering the object to be augmented using the method described above [3]. It computes visibility and lighting maps by matching the texture in the model image against that of the input image. The visibility map defines whether or not a pixel in the model image is visible or hidden in the input one. Considering a lighting map in addition to the visibility map allows to handle complex combinations of occlusion and lighting patterns. An example of visibility and lighting maps is shown in Fig. 4.

Since comparing the textures is sometime not enough to decide if the pixels are occluded or not, it imposes some spatial coherence on the visibility map. This is done by limiting the number of transitions between visible and occluded pixels.

3.2. Limitations

Considering the problem difficulty, the method described above gives good results, but the quality does not reached the standards for a large public application. Once again, the human eye instantly spots the mistakes done by the algorithm, in particular along the boundaries of the occluding objects. The simple spatial consistency ensured by the algorithm is definitively not sufficient to reach the same accuracy as a human, who can recognize the occluding object in Fig. 4 as a finger without any problem.

4. Conclusion

This paper tried to point out the limitations of the current Computer Vision techniques that prevent the implementation of mature Augmented Reality applications. Research has certainly reach the limits of what can be done with local low-level approaches, and a high-level "understanding" of the scene is now required from the computer to go further. Most of the recent advances in Computer Vision have been obtained mainly thanks to the introduction of Machine Learning techniques. The Object Recognition field in particular obtained impressive results, and as we tried to demonstrate in this paper, it is the most promising direction to solve the current limitations. We can only encourage researchers interested in Computer Vision for Augmented Reality to consider the Object Recognition field as a source of inspiration in the future.

5. Acknowledgments

The author would like to thank all the persons he had the pleasure to work with on Augmented Reality topics, in particular Pascal Fua, Julien Pilet, Mustafa Özuysal, Selim Benhimane, Stefan Hinterstoisser, Luca Vacchetti, Marie-Odile Berger, and Gilles Simon.

References

[1] S. Hinterstoisser, S. Benhimane, N. Navab, P. Fua, and V. Lepetit. Online learning of patch perspective rectification for efficient object detection. In *Conference on Computer Vision and Pattern Recognition*, 2008.

[2] F. Jurie and M. Dhome. Hyperplane approximation for template matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):996–100, July 2002.

[3] M. Ozuysal, P. Fua, and V. Lepetit. Fast Keypoint Recognition in Ten Lines of Code. In *Conference on Computer Vision and Pattern Recognition*, Minneapolis, MI, June 2007.

[4] J. Pilet, V. Lepetit, and P. Fua. Retexturing in the Presence of Complex Illuminations and Occlusions. In *International Symposium on Mixed and Augmented Reality*, Nov. 2007.

[5] G. Reitmayr and T. Drummond. Initialisation for visual tracking in urban environments. In *International Symposium on Mixed and Augmented Reality*, 2007.

[6] I. Skrypnyk and D. G. Lowe. Scene modelling, recognition and tracking with invariant image features. In *International Symposium on Mixed and Augmented Reality*, pages 110–119, Arlington, VA, Nov. 2004.