Application of augmented reality for supporting instrument service tasks

Toni Benussi, Zoran Kalafatic, Zeljka Mihajlovic

University of Zagreb, Faculty of Electrical Engineering and Computing, Unska 3, 10000 Zagreb, Croatia

Abstract

This paper describes a system for object detection and tracking implemented as an iPad application, which has been tested and successfully accomplishes its task. The system is based on finding the initial position of the object by matching features between a template and the device's picture. Tracking is achieved by using a pyramidal implementation of the iterative Lucas-Kanade algorithm. The implemented system was tested on two different instruments used for testing internal combustion engines. The paper discusses the possibilities offered by mobile devices, like the iPad, for the development of applications with computer vision and augmented reality elements and also describes the major problems that have been encountered on such platforms.

Keywords: computer vision, augmented reality, mobile devices, iPad, feature detection, tracking

1 Introduction

In the last few years the processing power of mobile phones and other hand-held devices, like tablets, have increased dramatically which allows more and more complex tasks to be achieved using such devices. One of the new possibilities is the use of mobile devices for real time processing of images and video sequences. The motivation for the research described in this paper was the desire to improve the installation and maintenance procedures of automated testbeds used for internal combustion engine testing. In this paper we tried to use a tablet device (iPad3) to detect and track known objects, specifically two peripheral devices of the automated testbed, using already established techniques which have been adapted and carefully parameterized to achieve real-time performances on a tablet. The goal is to overlap the important parts of the tracked object with useful information regarding such parts like technical documentation extracts, maintenance instructions and measured data.

The following section 2 shortly reviews the related work on this subject and after that the implemented algorithm is described, separately the initialization procedure and the tracking procedure. The results are presented in section 6 together with the encountered limitations and challenges present on mobile systems. The conclusion is given in section 7 along with potential future research directions.

2 Related work

An implementation of parallel tracking and mapping on camera phones is described in [1]. The goal in that paper is to create a map of the 3D environment from a series of images based on tracking of natural features. They track FAST [2] features while also simultaneously updating the feature map with new features from newly acquired images. Tracking is accomplished by predicting the most likely location of the features based on previous movement speed and direction and the results are refined by local search around the predicted location.

In [3] a feature tracking method is implemented. The method is based on SIFT descriptors proposed in [4], but substitutes the Difference-of-Gaussians (DoG) feature detector with the much faster corner detector from [2].

To match SIFT descriptors between the current image and the reference model the authors used Spill Trees described in [5]. To calculate the pose of the camera in relation to the tracked object they used Gauss-Newton iterative scheme to minimize the reprojection error to the image plane while taking into account the camera model with radial deformations.

3 Initialization

The goal of the initialization procedure of the algorithm described in this paper is to detect the instrument that will be tracked and determine it's position and orientation in the first frame from the iPad's embedded camera. The image acquired from the camera is first transformed into a grayscale image and the same is done for each subsequent frame. The iPad's camera suffers from high levels of noise, which is typical for embedded cameras. To mitigate this problem the grayscale image is processed with a Gaussian filter. The majority of the basic components and algorithms we used are implemented in the OpenCV library and written in C++, while the GUI was written in Objective C.

The next step is feature detection. Different algorithms were tested (FAST [2], SURF [6], SIFT [4]), but for the final implementation we used the detector *Good features to track* (GFTT) [7]. The available implementation of the SIFT detector was too slow and was immediately discarded, while the other detectors were tested and the results can be seen in table 1. The SURF detector was too slow for real time application. FAST has great performance and detects a lot of features compared to the other detectors, but even while using non-maxima suppression the detected features are tightly grouped together which can cause inaccurate computations of the homography later on. The GFTT detector has acceptable computation speed and a lot of parameters to refine the criteria for feature detection which was the primary reason for our choice.

Table 1: Comparison of feature detectors

	Duration iPad [ms]	Duration PC [ms]	Num. of features
FAST	36	7.5	1258
SURF	3413	670	632
GFTT	382	83	361

Feature descriptors are generated using the technique proposed in [6]. The specific objects we were trying to detect have many similar features on their surfaces, so to distinguish the features it was necessary to expand the obtained descriptors with scaled information about the position of the features on the object.

The same process of feature detection and description is performed offline on a set of template pictures of the object we want to track. The templates differ from each other by the angle and distance from which the object is observed. The position of the object on the templates is known beforehand. The descriptors of the features detected in the image from the live camera are matched to the features of each template using the FLANN [8] algorithm. The best matching template is chosen and 40 best feature matches between the image and the template are used to compute the homography matrix using the RANSAC [9] method. The homography matrix contains the information about the position of the object on the image in relation to its position on the template which is enough to start tracking the object and add virtual objects on top of it.

4 Tracking

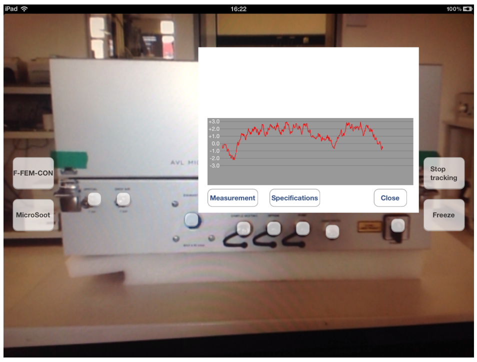
For tracking we used an iterative implementation of the Lucas-Kanade algorithm [10] that uses a pyramidal representation. We track 20 most prominent features obtained with [7]. Note that these features are not necessarily a subset of the 40 features used in the initialization as they are not chosen by the criteria of similarity with a template, but exclusively because they are very suitable for tracking.

The positions of the features on the current image and their counterparts on the previous image are used to compute the homography matrix between the two images. This solution to compute the homography between two close images has proven to be much more precise in comparison to computing the homography between the current image in the video sequence and the initial template, as it's easier to compute the homography for smaller changes of the viewpoint. Sometimes during the tracking procedure features are lost between to frames. Those features are excluded from the set used to compute the homography. To restore the lost features we use the homography computed from the successfully tracked features. Assuming that the homography is correct the lost features are restored to their actual position in the current image. This procedure allows us to maintain a constant number of features and to track the object for longer sessions.

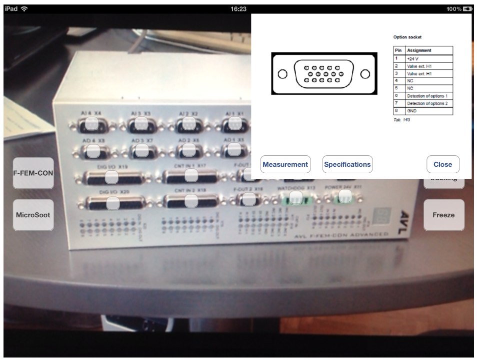
5 Virtual objects

By successfully tracking the object's position in the scene we are given the means to overlap parts of the objects with virtual objects containing useful information about them and create an augmented reality environment. The idea is to allow the user to interact with the components of the tracked device, in this case the MicroSoot smoke meter and the F-FEM-CON signal processor. The virtual elements allow the user to instantly get useful extracts from the manual or technical documentation specific to each component of the device or get real-time measurements and readings on the device from the central workstation which is connected to the iPad by a wireless network.

By knowing the position of the tracked device in the current image we can determine the exact position of each of the device's components and add the required information to the appropriate position in the image. Examples can be seen in figure 1.



(a) MicroSoot with measurement values



(b) F-FEM-CON with manual extract

Figure 1: Images of the devices with augmented reality elements

Table 2: Initialization evaluation

Device	Successful	Unsuccessful
MicroSoot	92.8%	7.2%
F-FEM-CON	87.75%	12.25%
Total	90.275%	9.725%

6 Results

"The two algorithm components, initialization and tracking, were separately tested. The initialization was tested on 90 pictures for each device and the result are shown in table 2. The results for the F-FEM-CON device are significantly worse then the Microsoot results because the F-FEM-CON has a large number of very similar features which causes mistakes during the matching procedure which yields a poorer homography.

Testing of the tracking procedure was performed on 3 video sequences of the Microsoot device with the total length of 172 seconds and on 4 video sequences of the F-FEM-CON device lasting a total of 176 seconds. We recorded the amount of time the devices were successfully tracked and also counted the events of tracker loosing the object and required to be reinitialized. The main reasons for the loss of the object are the high levels of motion blur present during camera movements and the narrow field-of-view of the camera which causes the object to leave the scene if larger movements are performed. The tracking evaluation results are presented in table 3. During the tracking procedure the system achieves 9 to 10 frames per second on average.

Table 3: Tracking testing results

	No. device lost	Tracking successfull
MicroSoot	6	80.8%
F-FEM-CON	5	84.1%
Total	11	82.5%

7 Conclusions and further work

In this paper we described a system for detecting and tracking objects with a handheld tablet device and added augmented reality components to the tracked object. The implemented system was tested and achieves a success rate of 90.275% in object detection and registration and the tracking procedure was successful in 82.5% of the total frames in the video sequences used for testing. The results are encouraging, however there are many issues still left to solve. The frame rate should be increased by parallelizing parts of the procedures, for example the tracking algorithm and the homography computation could be done in separate threads. Also the possibility to use the GPU of the mobile device for computation purposes could be worth investigating. An interesting solution to reduce the tracking problems caused by motion blur could be to use the embedded inertial sensors which are present in the iPad. Its measurements could be used to predict the camera movement in scenarios where the camera moves too fast for visual tracking to work correctly.

Acknowledgments

We thank the company AVL Graz for acquiring all the necessary equipment for our research and their employees for their ideas and suggestions that contributed to the realization of this paper.

References

[1] G. Klein and D. Murray. Parallel tracking and mapping on a camera phone. In *Mixed and Augmented Reality, 2009. ISMAR 2009. 8th IEEE International Symposium on*, pages 83–86. Ieee, 2009.

[2] E. Rosten and T. Drummond. Machine learning for high-speed corner detection. *Computer Vision–ECCV 2006*, pages 430–443, 2006.

[3] D. Wagner, G. Reitmayr, A. Mulloni, T. Drummond, and D. Schmalstieg. Pose tracking from natural features on mobile phones. In *Proceedings of the 7th IEEE/ACM International Symposium on Mixed and Augmented Reality*, pages 125–134. IEEE Computer Society, 2008.

[4] D.G. Lowe. Object recognition from local scale-invariant features. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 2, pages 1150–1157. Ieee, 1999.

[5] T. Liu, A.W. Moore, A. Gray, and K. Yang. An investigation of practical approximate nearest neighbor algorithms. *Advances in neural information processing systems*, 17:825–832, 2004.

[6] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. Speeded-up robust features (surf). *Computer Vision and Image Understanding*, 110(3):346–359, 2008.

[7] J. Shi and C. Tomasi. Good features to track. In *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on*, pages 593–600. IEEE, 1994.

[8] M. Muja and D.G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *International Conference on Computer Vision Theory and Applications (VISSAPP09)*, pages 331–340, 2009.

[9] Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[10] J.Y. Bouguet. Pyramidal implementation of the affine lucas kanade feature tracker description of the algorithm. *Intel Corporation*, 2001.