Universitat de les Illes Balears

escola politècnica superior

Treball Final de Carrera

ENGINYERIA INFORMÀTICA

Ponster, tu app de realidad aumentada

Álvaro Medina Ballester

Tutor
Ramón Mas Sansó

Escola Politècnica Superior
Universitat de les Illes Balears
Palma, September 7, 2014

CONTENTS

Contents **i**

1 Abstract **1**

2 Thanks **3**

3 Intro **5**

4 State of the art **7**
4.1 Object recognition 7
4.1.1 Template matching 8
4.1.2 Feature detection 8
4.2 Tracking 10

5 Technologies **11**

6 Development **13**

7 Conclusions **15**

CHAPTER 1

ABSTRACT

CHAPTER 2

THANKS

CHAPTER 3

INTRO

Augmented reality has become a very popular topic in the last five years. With the introduction of mobile smartphones, developers started to have the chance to develop applications using the powerful CPUs and GPUs of those devices.

CHAPTER 4

STATE OF THE ART

The techinique of mixing real world elements with virtual elements displayed on the screen of a device is what we call augmented reality. In the field of augmented reality, a lot of things have happened during the last years. The progress in the fields of computer vision and image processing have led to several new techniques of detection and tracking. This, combined with the increasing availability of powerful mobile devices, has enabled developers to build a plethora of high quality AR-based applications. Nowadays, modern mobile devices use integrated cameras, motion sensors and proximity sensors to make these AR-based experiences.

Augmented reality in mobile devices is slightly different from what can be seen in desktop environments. Although every year we have more powerful mobile smartphones (citation needed), processing and drawing into the device's screen is still an expensive operation in terms of computational cost. This is one of the reasons why cost-efficient computer vision algorithms and techniques have emerged in the past years.

Most of the augmented reality apps follow this behaviour:

- Get the input from the camera or a video.
- Search for an object of interest.
- Introduce our object into the scene, considering the camera or the input position.

In this chapter we are going to describe which are the different techniques that enables us to do them.

4.1 Object recognition

In order to provide an augmented reality experience, we have to know first which is the real world element that we are going to use as a reference to mix the real world input with our virtual elements. This reference can be from an image from the smartphone

camera to the user location. Ponster searches a particular image inside the camera input in order to draw the poster image above. In computer vision, this technique of searching an image and follow it along it's movement is usually referred as object tracking.

In computer vision there are a lot of object recognition techniques. In the development of Ponster, two of these techniques have been tested: template matching and feature-based detection. Detecting the image in a continuous input from the smartphone camera, taking account of scale, rotation and perspective differences, becomes an object tracking technique.

4.1.1 Template matching

Template matching consists of finding areas of an image that are similar to a provided template image. We have to provide a template image (the image that we want to look for) and compare it with the source image (the image in which we want to search). Template matching is also called area-based approach.

OpenCV provides a method to perform template matching with several methods, such as `SQDIFF` or `CCORR`. With the latest, `CCORR`, we use a correlation formula to check if the template is inside the image. Instead of applying a yes/no approximation, we can bring a positive match with a certain threshold.

Performing a template matching operation using OpenCV on mobile devices is fast enough to deliver a smooth 25/30fps-like detection. However, match template does not take account of scale, rotation and perspective invariance by itself. There are several approaches to bring invariation to match template. For instance, image pyramids are used to make match template scale and rotation invariant, but it is not part of the OpenCV match template function, although it provides some methods to implement image pyramids.

Match template has been tested during the development of Ponster. Also, a basic image pyramid system has been developed for scale-invariance, but match template has been discarded in favor of feature detection algorithms because rotation and scale invariance and perspective warp are required features.

4.1.2 Feature detection

A feature-based approach can be presented as a three step method[2]. First of all, we have to detect keypoints[1] (also called interest points) in the image. Usually, interest points are corners, blobs or T-junctions[2]. A good keypoint is a *repeatable* keypoint; if we can find the same keypoint under different conditions such as light difference or rotation, it's considered as a quality keypoint. The second step is to compute *descriptors* or feature vectors. These descriptors are represented as neighbourhoods of interest points. Assuming that feature detection makes sense when we have two images to compare, these two steps have to be performed on both images. Once we've done that, we have a group of descriptors for each image, and we have to compare them in order to *match* features. If the features of the source image are present in the input image, we can assume that the object has been detected. The matching is based on the distance between the feature vectors.

Usually, source images with enough keypoints are easier to detect than more uniform images. This is why it's better to select a good source image with many features and good contrast. As we've said before, in order to deliver a good augmented reality experience, we need to make our detection algorithm scale, rotation and perspective invariant. Feature detection techniques can be scaled invariant by extracting features that are invariant to scale, such as feature vectors computed from interest point neighbourhoods. For rotation invariance, algorithms can estimate the orientation of the keypoint.

There are plenty of feature-detection based algorithms, many of them based on Scale-Invariant Feature Transform, or SIFT. Cost efficiency is one of the most important features of these algorithms, as every new technique introduced tries to mantain robustness and reducing computation time. Robustness it's also very important, but less robust algorithms are also been developed in favour of reducing computation time. One good example is FAST[orb paper] keypoint detector, which is not rotation invariant. Next, we are going to describe the feature-detection algorithms tested on Ponster: SURF, FREAK and ORB.

Speeded-Up Robust Features - SURF

Speeded-Up Robust Features, also know as SURF, is a group of detector and descriptor introduced by Herbert Bay et al. SURF is faster and more robust than other alternatives like SIFT[2]. It's descriptors are rotation and scale invariant. Perspective transformations are also considered, but in lower order.

The keypoint detection in SURF uses a Hessian-matrix approximation. This use of integral images reduces computational cost in comparison with another interest point detection techniques such as Harris corner detection. Scale invariance is achieved by calculating integral image pyramids, but instead of reducing the image size, integral images allow SURF to upscale and build the pyramids more efficiently.

The SURF descriptor calculation is slightly based on SIFT. SURF descriptor describes the distribution of the intensity content within the interest point neighbourhood, which is similar to the gradient information used by SIFT. It is done in two steps, fixing a reproducible orientation based on information from a circular region around the keypoint, and then building a square region aligned to the calculated orientation. Once we have the descriptors, the last step is to perform the matching. Descriptors are compared only if they have the same type of contrast, allowing to perform a faster matching. In Ponster, two different matching algorithms have been tested, Brute-force Matcher and FLANN-based matching.

SURF is as robust as other alternatives such as SIFT, but it's faster to compute due to the use of integral images. It is rotational and scale-invariant, which is better than the template matching technique described before, but it's performance running on the device (iPhone 5, iOS 7.1.2) is not good enough to deliver a decent user experience, taking between 0.7 and 1 second to compute each image.

Features from Accelerated Segment Test - FAST

Oriented FAST and Rotated BRIEF - ORB

Brute-Force Matcher

Fast Library for Approximate Nearest Neighbors - FLANN

4.2 Tracking

CHAPTER 5

TECHNOLOGIES

CHAPTER 6

DEVELOPMENT

CHAPTER 7

CONCLUSIONS