ARTICLE IN PRESS

Pervasive and Mobile Computing ▮ (▮▮▮▮) ▮▮▮–▮▮▮

Contents lists available at ScienceDirect

Pervasive and Mobile Computing

journal homepage: www.elsevier.com/locate/pmc

Kernelized correlation tracker on smartphones

Danijel Peso [a,b,*], Alfred Nischwitz [a,**], Siegfried Ippisch [a], Paul Obermeier [b]

[a] Department of Computer Science and Mathematics, Munich University of Applied Sciences, 80335 Munich, Germany
[b] MBDA Deutschland GmbH, 86529 Schrobenhausen, Germany

ARTICLE INFO

Article history:
Received 11 October 2015
Received in revised form 8 May 2016
Accepted 25 June 2016
Available online xxxx

Keywords:
Smartphone
GPGPU
Correlation tracking
HOG features
Color features

ABSTRACT

This paper shows the implementation of a KC tracker (high-speed kernelized correlation tracker) on an Android smartphone. The image processing part is implemented with the Android-NDK in C/C++. Some parts of the tracking algorithm, which can be parallelized very well, are partitioned and calculated on the GPU with OpenGL ES and OpenCL. Other parts, such as the Discrete Fourier Transform (DFT), are calculated on the CPU (partly with the ARM-NEON features). With these hardware acceleration steps we could reach real-time performance (at least 20–30 FPS) on up-to-date smartphones, such as Samsung Galaxy S4, S5 or Google Nexus 5.

Beyond that, we present some new color features and compare their tracking quality to the HOG features using the KC tracker and show that their tracking quality is mostly superior compared to the HOG features.

If an object gets lost by the tracker which is the case e.g. if the object is totally hidden or outside the viewing range, there should be a possibility to perform a re-detection. In this paper, we show a basic approach to determine the tracking quality and search for the tracking object in the entire images of the subsequent video-frames.

Smartphones are getting more and more popular nowadays and some people cannot imagine to live without them anymore. With every smartphone generation, new features are getting available. As a consequence, the newest smartphones are getting more powerful with every generation. Consequently the computing power is also becoming more powerful, like the CPU (dual- and quad-core processors with higher clock frequencies), GPU and more and faster memory (some GBs). Moreover the SoCs[1] offer additional co-processors like DSPs.[2] For the Hexagon DSP in Qualcomm SoCs there is even a DSP-SDK publicly available.

Beyond that, smartphones offer a lot of sensors like a gyroscope, magnetometer, barometer, hygrometer, camera, microphone, infrared sensors, accelerometer, GPS, radio (Wi-Fi, bluetooth, mobile communications) and many more. All offered hardware is easily programmable via Android-SDK (Java) and -NDK (C++), drivers for all sensors are ready to go.

Above all of these functionalities smartphones are lightweight and pretty cheap compared to the offered hardware. As a consequence, smartphones are an ideal "head" of autonomous systems such as robots and UAVs.[3] The main features of

* Corresponding author at: Department of Computer Science and Mathematics, Munich University of Applied Sciences, 80335 Munich, Germany.
** Corresponding author.
E-mail addresses: dpeso@web.de (D. Peso), nischwitz@cs.hm.edu (A. Nischwitz), siegfried.ippisch@gmail.com (S. Ippisch), paul.obermeier@mbda-systems.de (P. Obermeier).

1 System on Chip, i.e. an integrated circuit where CPU, GPU, DSP etc. are contained.
2 Digital Signal Processor is a microprocessor designed for signal processing.
3 Unmanned Aerial Vehicle.

http://dx.doi.org/10.1016/j.pmcj.2016.06.013
1574-1192/

a "head" are the brain with lots of memory and computing power (CPU, GPU and DSP of a smartphone) and the senses (sensors). One important task for autonomous systems is visual tracking of moving objects.

Visual Tracking is used in different areas like man-machine interaction, virtual reality, robotics, medicine or video surveillance. In the past decades, more robust tracking methods were developed. The goal of visual tracking is to find/detect the target(s), which are supposed to be tracked in subsequent frames and to estimate their positions and orientations. The successful tracking is the core task, but at the same time a big challenge, when there are bad conditions like noisy images, small resolutions of images, occlusions, deformations, rotation of the object to be tracked, background-clutter, changing light conditions and real-time requirements.

In this paper, a model-less tracker (KC tracker) based on the following two papers [1,2] is implemented to run on smartphones. To accelerate the computations, the GPU of the SoC is used for certain computation parts, if OpenGL ES 2.0 and/or 3.0 and/or OpenCL are supported by the hardware and OS. The tracker performs its calculations in the Fourier space. Thus a fast FFT implementation, respectively FFT-library is necessary, running on an ARM-based smartphone. To accelerate the FFT transforms, besides the GPU, ARM-NEON is an option which is used here. The tracker is running on different Android smartphones (Samsung Galaxy S4, S5 and the Google Nexus 5), which all contain different Qualcomm-SoCs, to compare the performances with each other.

Moreover, this work compares the tracking quality of some basic approaches to color features with the HOG features, as the used tracker can operate with different features, e.g. raw features (pixels), HOG features and so on. If an object gets lost, one could search for the object over the entire image. Therefore, we use a simple way to determine a bad tracking quality and re-detect the object in the subsequent frames afterwards.

1. State-Of-The-Art

1.1. Tracker benchmark

Ways of measuring the tracking performance are explained in [3], which describes three ways of measuring the tracking accuracy. One method is the OPE (One-Pass Evaluation), which is used in our work to compare tracking performances. This type of precision plot measures the average precision for a video sequence from the beginning to the end. With it, the percentages of frames of a sequence are measured, which do not exceed certain error thresholds. The location error threshold is the Euclidean distance between the calculated tracking position and the ground truth data of a frame. Furthermore, the tracking performance over 50 different video sequences is measured for 29 different tracking algorithms in the mentioned benchmark paper [3]. For comparison and ranking of different trackers, an error threshold of 20 pixels is used. In the following the best 2 out of these 29 tracking algorithms are explained in more detail.

1.2. Struck

Struck ranks currently among one of the best model-less state-of-the-art trackers, however it is also one of the slowest trackers as the computation time is high. A lot of tracking-by-detection approaches require an intermediate step, where the labeling takes place and where binary labels are used. The publication where the Struck tracker is presented [4] considers this as a problem, because traditional labelers use a "transformation similarity function", which is rather selected by intuition and heuristics. Instead of training a classifier, in [4] a probability function $f : x \rightarrow y$ is learned, to estimate the transformations between two frames directly. For learning, the Kernelized Structured Output SVM framework is used.

1.3. Kernelized correlation tracking algorithms

We use the generic term "Kernelized Correlation Tracker" (KC tracker) for a class of tracking algorithms introduced by Henriques et al. in [1] and [2], that exploit the circulant structure of tracking-by-detection with kernelized correlation filter algorithms. The first version [1] (called csk tracker in the benchmark paper [3]) shows, that the subwindows which are used for tracking have a circulant structure and how one can use this knowledge efficiently in the Fourier domain by diagonalization. For the learning/regression, KRLS (Kernelized Regularized Least Squares) is used. There are several types of kernels, like linear or Gaussian kernels, which can be used and still preserve the circulant structure of the matrices.

The second version [2] explains the procedure in more detail and with proofs of the used formulas. Furthermore, the use of features (e.g. HOG features and others are possible) within the tracker is introduced which requires multi-channel correlation filters.

In comparison to other model-less trackers, this tracker reaches a very good tracking quality and still has a fast runtime. Other trackers with similar tracking qualities, which we could find, are significantly slower. With the use of HOG features it even outperforms other state-of-the-art trackers, though it is still much faster than other trackers with a comparable quality.

The algorithm consists of the following steps:

1. At the beginning, a start frame with subwindow size and subwindow position, which should contain the target, is set by the user.

Please cite this article in press as: D. Peso, et al., Kernelized correlation tracker on smartphones, Pervasive and Mobile Computing (2016), http://dx.doi.org/10.1016/j.pmcj.2016.06.013