Efficient Extraction of Robust Image Features on Mobile Devices

Wei-Chao Chen  Yingen Xiong  Jiang Gao  Natasha Gelfand  Radek Grzeszczuk*

Nokia Research Center, Palo Alto

Recent convergence of imaging sensors and general purpose processors on mobile phones creates an opportunity for a new class of augmented reality applications. Robust image feature extraction is a crucial enabler of this type of systems. In this article, we discuss an efficient mobile phone implementation of a state-of-the-art algorithm for computing robust image features called SURF. We implement several improvements to the basic algorithm that significantly improve its performance and reduce its memory footprint making the use of this algorithm on the mobile phone practical. Our prototype implementation has been applied to several practical applications such as image search, object recognition and augmented reality applications.

1 Introduction

Integration of cameras and general-purpose processors in mobile phones signals the dawn of a new class of mobile computation devices. In the field of computer vision and graphics, we witness a variety of applications utilizing the mobile phone platforms, including MARA [2] and Lincoln [7].

One particularly appealing, but challenging, aspect of mobile augmented reality is feature tracking for augmentation of the real-world imagery with overlay information similar to Skrypnyk and Lowe [6]. An important component of such application is the computation of robust, scale-invariant feature descriptors [4, 1]. Figure 1 shows a typical image matching algorithm pipeline using feature descriptors.

These algorithms tend to be very computationally intensive even for modern desktop PCs. Mobile devices, despite their rapid advances, will likely not match the desktop PC performances in the near future. Naturally, we have an option of sending a query image to the server and computing its features there, but the wireless network latency and slow uplink bandwidth pose severe constraints on real-time AR applications. It is therefore an interesting challenge to optimize these algorithms for both space and performance to make their execution on a mobile platform efficient.

Our Contributions. We have chosen the SURF algorithm [1] as the basis of our implementation because of its favorable computational characteristics and its state-of-the-art matching performance. We then implemented and optimized this algorithm on a mobile phone. Our implementation is on average 30% faster and uses half as much memory.

2 Platform Considerations

We target a mobile phone platform that uses Texas Instrument’s OMAP 2 application processor architecture which integrates an ARM11 core, an image/video accelerator, a DSP and a PowerVR graphics core, among others. The operating system of choice for our mobile device is Symbian OS. The amount of memory available to each thread running on a mobile device is typically fairly limited to ensure system stability. Hence, a careful consideration must be given to a mobile phone implementation of an algorithm to ensure good performance.

3 Implementation

The SURF algorithm [1] consists of three major steps: interest point extraction, repeatable angle computation and descriptor computation. The interest point extraction step starts with computing the determinant of the Hessian matrix and extracting local maxima. The Hessian matrix computation is approximated with a combination of Haar basis filters in successively larger levels. Therefore this step is \(O(mn\log_2(\max(m,n)))\) for a \(m \times n\) size image. Each extracted interest point is further improved by quadratic localization.

After the interest points and their scales are obtained, a repeatable angle is extracted for each interest point prior to computing the feature descriptor. This step computes the angle of the gradients surrounding the interest point and the maximum angular response is chosen as the direction of the feature. This direction is then used to create a rotated square around the interest point, and regularly sampled gradients within this template are combined per grid location to form the final descriptor. Because both of these steps require processing image footprints proportional to the interest point scale, an efficient sampling algorithms can speed up significantly in these two steps. Finally, we evaluated the quality of our implementation and adjust the parameters in the algorithm using the benchmarks from the paper by Mikolajczyk et al [5].

3.1 Our Implementation

In the first two steps of the algorithm we use integral image for efficient Haar transform computation similar to [1]. It takes only two floating operations per pixel to transform from a regular image to integral image in-place, therefore we only store either the original or integral image, and convert them back and forth as needed.

Interest point detection. This step involves computing Hessian determinant value at every location \((x, y)\) on the scale space \(s\),
followed by a $3 \times 3 \times 3$ local maximum extraction and quadratic localization. Because the minimum scale uses a Gaussian at $\sigma = 1.2$, by setting the cutoff frequency at 50% of the maximum amplitude, the Nyquist sampling rate is equivalent to $1/3\cdot2$ of the original image resolution. We choose $2 \times$ sub-sampling rate at the minimum scale level, and store only three scale levels necessary for the local maximum extraction. This consumes $3 \times 1/4 = 75\%$ of the input image.

Repeattable Angle Computation. We generate the Gaussian filter lookup table to reduce the floating point computation requirement. We also used an efficient arctan approximation to speed up the angle computation process. This approximation is only used in the angle binning process and, as a result, it yields almost no change to the final extracted angle. Overall, this stage is extremely efficient and incurs only minor overhead ($<10\%$) compared to upright-SURF.

Descriptor Extraction. In this step, we need to compute image gradient regularly sampled near the interest point. The sampling grid is rotated to the angle computed from the previous step. To speed up the resampling process, we pre-compute mipmapfs using the round-up algorithm in [3]. One of the main advantages is we can resample at the proper scale prior to computing the feature descriptors, and as a result each pair of the $(dx,dy)$ Haar transform computation operates in the downsampled $2 \times 2$ pixel grid regardless of the scale of the interest point. This approach, however, requires 33% memory overhead in addition to the original image. It also uses the original input image which is reverted in-place from the integral image.

4 Results
First we compare our PC feature descriptor implementation against the published implementation from [1]. Figure 2 shows the runtime on a laptop with Intel Core Duo T2500 processor operating at 1.8Ghz. We use the test images from the paper by Mikolajczyk et al. [5] for our experiments. Each point in this figure represents the execution time over four scale octaves and 2x initial subsampling during interest point localization. Because the descriptor computation time is fairly proportional to the number of detected features while the interest point extraction is dependent of the image resolution, we run each test image over different detection thresholds and compute the feature density (namely, detected features for every image pixel) on the x-axis. On average, we achieve approximately 30% speedup over [1], using double precision floating point numbers in both cases.

Figure 2: Actual runtime of both our and implementation from [1] running at different detection thresholds.

Since we are not aware of any implementation of the SURF algorithm for a mobile phone, we compare the ratio of our algorithm running on the phone versus running on the PC, as shown in Figure 3. The test images are randomly chosen from [5]. The phone version, running on the Nokia N95 smart phone, shows on average 22x slowdown compared to the PC. Since the ratio is relatively high, we believe a further improvement in the mobile phone performance is possible by taking advantage of the special image processing instructions available on the embedded CPU and by vectorizing parts of the code for the built-in floating point SIMD unit. We leave this as future work. We are also looking at developing a new, robust feature extraction algorithm that would be even more efficient than the current implementation thanks to a more direct mapping to the mobile platform.

5 Conclusion
We have described a mobile phone implementation of a state-of-the-art robust feature descriptor algorithm. We have achieved a significant improvement in performance over the original implementation, which allow us to explore many interesting new research direction, which before were only possible with bulky PC/camera systems and can now run on a small mobile device.

To our knowledge, our implementation is the first of its kind on a mobile phone. We are excited about the new opportunities this creates for mobile augmented reality applications and are planning to continue improving the results published here. We hope to release the code to the public in the near future.

References