Performance Evaluation of BRISK Algorithm on Mobile Devices

Alexander Gularte, Camila Thomasi, Rodrigo de Bem and Diana Adamatti
Centro de Ciências Computacionais (C3), Universidade Federal do Rio Grande (FURG),
Av. Itália, km 8, 96203-900, Rio Grande, RS, Brazil

Keywords: Performance Evaluation, Local Features Extraction, Mobile Imaging.

Abstract: The great number of researches about local features extraction algorithms in the last years, allied to the popularization of mobile devices, makes desirable efficient and accurate algorithms suitable to run on such devices. Despite this, there are few approaches adequate to run efficiently on the complexity-, cost- and power-constrained mobile environments. The main objective of this work is to evaluate the performance of the recently proposed BRISK algorithm on mobile devices. In this way, a mobile implementation, named M-BRISK, is proposed. Some implementation strategies are considered and successful applied to execute the algorithm in a real-world mobile device. As evaluation criterion repeatability, recall, precision and running time metrics are used, as well as the comparison with the classical well established algorithm SURF and also with the more recently proposed ORB. The results confirm that proposed mobile implementation of BRISK (M-BRISK) performs well and it is adequate to mobile devices.

1 INTRODUCTION

The widespread use of digital cameras is responsible for the generation of millions of images daily. Mobile digital devices, such as mobile phones and tablet PCs, extend the digital still cameras functionalities, allowing real-time exchange of images through their communication modems. These kind of devices, usually capable of capturing images and videos, are nowadays available to millions of people over the world. This fact makes them important tools for image acquisition, allowing anyone to easily capture images of products, advertisements, people’s faces and bodies, pieces of art or any other visual data from the environment where he or she is inserted. Extracting useful information from all these data, through efficient computational algorithms, is an important goal of the Computer Vison research community.

The use of distinctive visual features to accomplish this goal has been highly explored in the last decade and applied to many problems such as content-based image retrieval (CBIR) (Smeulders et al., 2000), markerless augmented reality (MAR) (Skrypnyk and Lowe, 2004) and 3D scene reconstruction (Brown and Lowe, 2005). When a feature extraction algorithm employs an unique descriptor to an entire image it is called global. Otherwise, if it uses descriptors to many image points, it is called local. The local feature extraction algorithms are widely used because their invariance to many visual transformations. These extractors are able to detect salient regions into an image, which are called keypoints. Each keypoint is described by a feature vector, that represents numerically the image properties on the neighborhood of such point. These sets of keypoints descriptors can be used to perform matching between images, through the calculation of similarity metrics among them.

Despite the great profusion of researches about features extraction algorithms in the last years, there are few approaches suitable to run efficiently into mobile devices. Some algorithms, developed to have the best performance into unconstrained environments, do not perform well in the complexity-, cost- and power-constrained environment of such devices (Budagavi, 2006).

In this context, the main contribution of this work is to evaluate the performance of the recently proposed BRISK algorithm (Leutenegger et al., 2011), a local feature extractor, on a mobile device. To achieve it, we did a consistent analysis of the BRISK method, as well as the identification of its critical points concerning the computational cost. Some implementation strategies were considered and successful applied to execute the algorithm in a real-world mobile device. As evaluation criterion repeatability, recall, precision and running time metrics (Mikolajczyk and Schmid, 2005) were used, as well as the comparison...
with the classical well established algorithm SURF 
(Bay et al., 2006) and also with the more recently pro-
posed ORB (Rublee et al., 2011). The results confirm 
the hypothesis that proposed mobile implementation 
of BRISK (M-BRISK) performs well and it is ade-
quated to mobile devices.

This paper is organized as follows. In Section 2 
we present the main considered algorithms for local 
features extraction. In Section 3 the BRISK method 
is reviewed, while their mobile implementation is pre-
sented in Section 4. The experiments and results are 
shown in Section 5. And finally, Section 6 presents 
the conclusions and future work.

2 LOCAL FEATURES 
DETECTION AND 
DESCRIPTION

Lowe (Lowe, 2004) proposed the Scale Invariant Fea-
ture Transform (SIFT) detector and descriptor algo-
rithm, for extracting high distinctive features from 
images. The SIFT method, one of the most reliable 
features extractors, is invariant to scale and rotation 
transformations, and partially invariant to brightness 
and viewpoint changes. This algorithm is composed 
by four main stages: scale-space extrema detection, 
keypoints localization, keypoints orientation and key-
points description.

Due to the high dimensionality of the SIFT de-
scriptor and its poor computing performance, other 
less time-consuming alternatives began to be pro-
posed in the literature, with the purpose of enabling 
their use in real-time applications. In PCA-SIFT (Ke 
and Sukthankar, 2004) algorithm, the Principal Com-
ponents Analysis (PCA) was applied to decrease the 
dimension of SIFT descriptors and to improve the 
matching time.

Bay et al. (Bay et al., 2006) presented a novel ap-
proach, partly based on SIFT, called Speed Up Robust 
Features (SURF). For some cases, the SURF algo-
rithm is more robust than SIFT, taking less computing 
time. SURF employs integral images to build a fast 
detection method called Fast-Hessian. The descrip-
tion is based in a first order Haar Wavelet distribu-
tion. The SURF proves to be a very efficient method, 
however it has limited performance when compared 
to recent binary description approaches.

Calonder et al. (Calonder et al., 2010) built a bi-
ary descriptor, called Binary Robust Independent El-
ementary Features (BRIEF), with low dimensionality 
and more suitable for real-time applications. In 
this approach, the descriptors are formed by binary strings 
which makes comparisons more efficient. However, 
despite the high performance, the algorithm suffers in 
terms of robustness and reliability, because it has low 
tolerance in terms of distortion and image transforma-
tions, such as scale and rotation.

In order to overcome the BRIEF limitations, in 
terms of rotation invariance, the Oriented FAST 
and Rotated BRIEF (ORB) (Rublee et al., 2011) algorithm 
was proposed. This method uses Features from Ac-
celerated Segment Test (FAST) (Rosten and Drum-
mond, 2006) detector to provide a complete detection 
and description algorithm. It has a good robustness 
and a good performance, even concerning mobile de-
vices.

Leutenegger et al. (Leutenegger et al., 2011) pro-
posed the Binary Robust Invariant Scalable Keypoints 
(BRISK), a new approach that joins the fast and ef-
cient Adaptive and Generic Accelerated Segment 
Test (AGAST) (Mair et al., 2010) detector to a bi-
ary descriptor inspired by BRIEF. This algorithm, 
that uses a sampling pattern resembling the one used 
in DAISY dense descriptor (Tola et al., 2010), is in-
variant to many transformations, such as scale and ro-
tation. BRISK allies high-quality descriptor to low 
computational requirements, what makes it proper to 
real-time applications.

3 BRISK REVISITED

The BRISK method can be divided in three steps: 
keypoints detection, keypoints description and descriptors 
matching\(^1\). These stages are executed se-
quently.

3.1 Keypoints Detection

It is performed at the first moment. The goal at this 
step is to identify salient points in the image that, ide-
ally, could be uniquely differentiated from any other 
point. To do so, these points must be searched across 
the image and scale dimensions using a saliency cri-
terion (Leutenegger et al., 2011). The search through 
scale space is fundamental to achieve scale invariance 
and it is realized in BRISK by the use of a pyramid 
of images. This pyramid is formed by many layers 
which correspond to resamples of the original image. 
In BRISK method, these layers are divided in \(n\) oc-
taves \(c_i\) and \(n\) intra-octaves \(d_i\), with \(i = \{0, 1, \ldots, n\}\) 
and usually \(n = 4\). The intra-octaves \(d_i\) are located 
between octaves \(c_i\) and \(c_{i+1}\). These intra-octaves

\(^1\)The matching step can also be seen as an independent 
stage.
and octaves are formed by the original image subsampled by factors $\frac{1}{2}$ (half-sampling) and $\frac{2}{3}$ (two-third-sampling). The original image is progressively half-sampled to form the octaves, while the intra-octaves are formed by progressively half-sampling of the first intra-octave $d_0$, which in turn is composed by a two-third-sampling of the original image.

After the octaves and intra-octaves calculation, the AGAST corner detector (Mair et al., 2010) is applied to each one of them. All the points of each layer will be evaluated as a keypoint candidate, and to be elected a point must be salient among their intra-layer and inter-layers neighbors. It means that each point is compared with its neighbors in the same layer and also with those in the above and bellow layers. After the analysis of all octaves and intra-octaves, the detection stages finally ends, producing a set of keypoints with space and scale coordinates, what means that each one of them can be exactly located.

3.2 Keypoints Description

In this stage, around each keypoint, which has its co-ordinates found in the previous step, a sampling pattern is positioned. The BRISK algorithm uses a sampling pattern similar to the one in DAISY descriptor (Tola et al., 2010). However, it is important to note that the use of this pattern is quite different. The pattern relies on 60 equally spaced points located in four concentric rings. In order to produce a scale and rotation invariant description, the sampling pattern is exactly scaled and rotated according to each keypoint. The BRISK descriptor is composed as a binary string of length 512. This string concatenates the results of simple brightness comparisons tests between each keypoint and its 60 neighbors in the pattern (Leutenegger et al., 2011). This approach was inspired by the BRIEF method (Calonder et al., 2010).

3.3 Matching

Finally, to perform the comparison between two or more keypoint descriptors, the Hamming distance is used. This distance measures the number of different bits between two binary strings and it represents the degree of inequality of the descriptors being compared.

4 M-BRISK

This section presents the mobile implementation of BRISK algorithm and the main aspects concerning to the environment where it was developed and tested. This mobile implementation is called M-BRISK (Mobile-BRISK).

4.1 Implementation

In order to implement the algorithm to be executed into a mobile device, some issues had to be overcome. The first of them was the fact that SSE2 and SSE3 instructions (Intel, 2006), the Intel’s SIMD instructions set (Flynn, 1972), were employed by Leutenegger et al. (Leutenegger et al., 2011) in their original implementation of BRISK. It was used as a crucial resource to improve the BRISK performance at critical time-consuming routines.

SIMD instructions are available in ARM architecture (ARM, 2012b), the most popular architecture of mobile devices processors, just since the ARMv7 version, through the NEON instructions (ARM, 2012a). As these instructions are not available into all devices, specially into the low-end ones, it was decided not to use them in the present implementation. This choice allows M-BRISK to be a more generic implementation, avoiding the dependence of specific mobile processors and instructions sets. This approach also constitutes a baseline assessment of the algorithm once it is being evaluated in a most ordinary condition.

The sub-sampling routines, originally implemented trough the SSE instructions, was replaced by a bilinear interpolation function implemented using the OpenCV library (OpenCV, 2011). This approach showed to be efficient and applicable to a mobile environment while it kept robustness.

Another critical point is the sampling pattern routine. In the original BRISK implementation, as a manner to obtain this pattern efficiently, 65536 samples are generated to each one of the 60 pattern points during the algorithm initialization and stored in a lookup table. These samples corresponds to all possible discretized values of scale and rotation. There are 64 and 1024 possible values to each dimension, respectively. The goal of this approach is to save running time in the description construction of each keypoint. However, such process is so time-consuming that its use is prohibitive on mobile devices. Thus, to avoid this problem, we chose to previously save a binary representation of the look-up table in mobile device memory card. Doing so, at the time of descriptors construction the table is already completely loaded into device memory.

Finally, during the matching routine, the SSE instructions were also used to improve the performance.

---

2The original BRISK implementation is available in: http://www.asl.ethz.ch/people/lestefan/personal/BRISK
of original BRISK implementation. These instructions were used in this stage to optimize the calculation of the Hamming distance, employed to compare two keypoint descriptors. In M-BRISK the Hamming distance was also implemented through the OpenCV library. Unfortunately this change represented considerable addition in the running time of matching.

4.2 Mobile Execution Environment

The mobile platform chosen to support the implementation of M-BRISK algorithm was the Android (Google, 2012). This decision was motivated mainly because it is a well documented open development platform. Another distinguishing feature of Android is its ability to run very efficiently C and C++ libraries, allowing the reuse of previously implemented code. In order to execute these libraries into Android, the Native Development Kit (NDK) (Google, 2011) was used. Since Android applications are written in Java, to enable interfacing between the different languages, the NDK relies on the Java Native Interface (JNI) (Liang, 2003). The Android 2.2 version was used.

The employed device was a low-end mobile phone with Mediatek MTK6516 416MHz ARMv5 processor, with 256MB of RAM memory. As mentioned earlier, the NEON instructions are not available on this processor architecture version.

5 EXPERIMENTS AND RESULTS

Leutenegger et al. (Leutenegger et al., 2011) compared BRISK with SIFT and SURF algorithms, showing the high performance of the former. The purpose of the present experiments was to evaluate the performance of M-BRISK. To accomplish this, the mobile implementation is compared with the original BRISK implementation. Tests were performed to compare the repeatability rate and to measure the relation between execution time in the steps of detection, description and matching of the both approaches. Moreover, M-BRISK it is compared with SURF and ORB using the recall and precision metrics and also on a real mobile device, used to compare the running time of such algorithms.

5.1 Image Dataset

The image dataset[^3] used to perform the experiments is a well established benchmark, largely employed to evaluate many methods in this field (Calonder et al., 2010), (Mikolajczyk and Schmid, 2005), including the BRISK algorithm (Leutenegger et al., 2011). The transformations covered by the dataset, shown in the Figure 1, are: viewpoint change (Graffiti and Wall), blur (Bikes and Trees), brightness change (Leuven), zoom and rotation (Boat) and JPEG compression (Ubc).

5.2 Repeatability Test

The repeatability test was used to identify the ability of the detector to find the same keypoint into distinct images modified by some kind of transformation. The goal of this test is to evaluate if the modifications in the sub-sampling methods (half-sampling and two-third-sampling) affected the quality of the detection in the M-BRISK implementation.

The results in Figure 2 show the repeatability for viewpoint change (Boat images) and zoom and rotation transformations (Wall images). The percentual of repeatability measured for each algorithm is shown on top of each bar. Based in the values, we can conclude that M-BRISK has repeatability similar to the BRISK, proving that the mobile implementation of the detection step did not degrade the quality of the algorithm. In some cases, M-BRISK has even slightly better results than BRISK.

5.3 Recall × 1-Precision Evaluation

Recall and 1-precision are two relevant metrics used to show the effectiveness of a descriptor method. Recall measures the proportion of correct positives matches, considering all the possible correct positive matches between two images. While 1-precision
5.4 Running Time Test

These tests compare the time variations in the different stages of BRISK and M-BRISK implementations. Moreover, the comparison between the mobile device running time of M-BRISK, SURF and ORB are shown.

5.4.1 Distribution of Running Time between BRISK and M-BRISK Stages

In Figure 4, the mean time per point spent in each step of BRISK and M-BRISK is presented. The purpose is to analyse the relation between the running time of each algorithm step aiming to evaluate the effects of the modifications over BRISK. The tests were performed using 100 samples of images for the sets Graffiti, Boat and Leuven. The execution was hosted in a computer with a Core2Duo T8100 2.10Ghz processor, 3GB of RAM memory and Windows 7 (64-bits) operating system.

The mean times and the standard deviations are showed at the top of the bars. As can be observed, in the average, M-BRISK had performances in the detection and matching stages, respectively, around 15% and 75% worst than BRISK. Even without the use of SSE instructions, what was the reason of the M-BRISK worst results, the time consumed in the detection stage was not affected so significantly. In the description stage the performance of both implementations was similar.

Another important difference between the two approaches is concerning the look-up table usage. While it is calculated during the execution in BRISK, in M-BRISK the look-up table is calculated previously. The mean time consumed in the BRISK look-up table calculation is 1.269s, while the mean time spent to load it from memory in M-BRISK is 0.088s. The latter amount of time corresponds to less than 10% of the former.

5.4.2 Running Time Test on a Mobile Device

The purpose of this test is to provide a reference for running times of M-BRISK, SURF and ORB in a real mobile device. These tests are performed using the images Graffiti 1-3 with 50 samples.

Tables 1 and 2 show the execution running times in the mobile device. The times in the tables represent an average of all running time samples with their respective standard deviation between parenthesis. As we can observe, the M-BRISK is about twice faster than ORB and more than 20 times faster than SURF in detection and description steps. In the matching
Figure 4: Comparison between the running time of BRISK and M-BRISK steps. The mean amount of time spent per point in each step is shown at the top of the bars, as well as the standard deviations in parenthesis.

step, the time difference is not so significant comparing ORB and M-BRISK. However, SURF had worst results than M-BRISK in all execution steps.

The Table 1 shows the detection and description times. The time per point represents the average time for detection and description for each keypoint. This value represents a fairness comparison between the algorithms. The Table 2 shows the matching times, including the average time spent during each descriptor comparison. The serial implementation of Hamming distance decreased the matching performance, however remained very fast when compared with SURF descriptors matching. Even compared with ORB, M-BRISK implementation still demonstrates pretty competitive timings.

Table 1: Detection and description timings for the Graffiti image 1.

<table>
<thead>
<tr>
<th></th>
<th>SURF</th>
<th>ORB</th>
<th>M-BRISK</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of points</td>
<td>3201</td>
<td>1000</td>
<td>1957</td>
</tr>
<tr>
<td>Detection time (s)</td>
<td>65.87 (1.56)</td>
<td>3.17 (1.13)</td>
<td>1.83 (1.81)</td>
</tr>
<tr>
<td>Description time (s)</td>
<td>140.66 (0.50)</td>
<td>2.85 (1.25)</td>
<td>3.59 (3.53)</td>
</tr>
<tr>
<td>Total time (s)</td>
<td>206.54 (1.71)</td>
<td>6.02 (1.89)</td>
<td>5.42 (5.35)</td>
</tr>
<tr>
<td>Time per point (ms)</td>
<td>64.52 (0.53)</td>
<td>6.02 (1.89)</td>
<td>2.77 (2.78)</td>
</tr>
</tbody>
</table>

Table 2: Matching timings for the Graffiti images 1 and 3.

<table>
<thead>
<tr>
<th></th>
<th>SURF</th>
<th>ORB</th>
<th>M-BRISK</th>
</tr>
</thead>
<tbody>
<tr>
<td>Points in first image</td>
<td>3820</td>
<td>1000</td>
<td>2529</td>
</tr>
<tr>
<td>Time (s)</td>
<td>570.66 (17.90)</td>
<td>5.99 (2.44)</td>
<td>26.59 (9.66)</td>
</tr>
<tr>
<td>Time per comparison (ms)</td>
<td>30.31 (1.46)</td>
<td>5.99 (2.44)</td>
<td>5.37 (1.95)</td>
</tr>
</tbody>
</table>

6 CONCLUSIONS

The wide use of local features extractors applied to many computer vision problems, allied to the popularization of mobile devices, makes desirable efficient and accurate algorithms suitable to run on such devices. These algorithms must join robustness and low computational cost, and still there are few methods efficiently adequate to the constrained mobile environments. Thus, the main objective of this work was to evaluate the performance of the recently proposed BRISK algorithm on a mobile device. To do so, a mobile implementation, named M-BRISK, was proposed.

M-BRISK is focused to run in the most common ARM processors architecture. The implementation successfully overcame some mobile environment constraints, as processing power and limited instructions set, confirmed through the analysis of the obtained results. These showed that M-BRISK is so efficient and accurate as the original BRISK implementation, which presents excellent results even when compared with the well established SIFT and SURF. Moreover, M-BRISK presented a overall running time approximately twice smaller than ORB and 20 times smaller than SURF, the other mobile approaches evaluated. Thus, it is possible to point M-BRISK as a very suitable approach for detection and description on mobile applications.

In future works, a version of M-BRISK with NEON instructions will be tested against the serial version. Also other parallel development paradigms, such as GPGPU processing, supported by the most advance mobile devices, must be considered. Moreover, a detailed analysis of M-BRISK computational complexity is desirable.

REFERENCES

Performance Evaluation of BRISK Algorithm on Mobile Devices


