International Journal of Advanced Computer Research (ISSN (print): 2249-7277 ISSN (online): 2277-7970)
Volume-3 Number-3 Issue-11 September-2013

Object recognition with ORB and its Implementation on FPGA

A.V.Kulkarni[1], J.S.Jagtap[2], V.K.Harpale[3]

Abstract

Object recognition and tracking has a wide spectrum of promising applications. Feature-based algorithms are well-suited for such operations like Speeded Up Robust Features (SURF), SIFT (Scale-invariant feature transform), ORB (Oriented FAST and Rotated BRIEF) algorithm has been proved to achieve optimal results. ORB algorithm builds on the well-known FAST key point detector and the recently-developed BRIEF descriptor. This paper gives an overview of a general methods of object recognition and significance of ORB over SIFT and SURF in different cases. This paper also provides an idea to implement ORB algorithm on FPGA to increase the execution speed by utilizing the reconfigurable nature and pipelining of the FPGA.

Keyword

SIFT, SURF, ORB, FAST, and BRIEF.

1. Introduction

Feature-based image matching is an important aspect in many computer based applications, such as object recognition, images stitching, structure-from-motion and 3D stereo reconstruction [2]. These applications require often real-time performance. Feature-based algorithms are well-suited for such operations. Different algorithms are used for image processing like Scale-invariant feature transform (SIFT), Speeded Up Robust Features (SURF), Oriented FAST and Rotated BRIEF (ORB).

ORB is a very fast binary descriptor based on BRIEF, which is rotation invariant and resistant to noise. It can be demonstrated through experiments how ORB is at two orders of magnitude faster than SIFT, while performing as well in many situations. The efficiency is tested on several real-world applications, including object detection and patch-tracking on a smart phone.

ORB builds on the well-known FAST keypoint detector and the recently-developed BRIEF descriptor; for this reason we call it ORB (Oriented FAST and Rotated BRIEF). Both these techniques are attractive because of their good performance and low cost. ORB includes the addition of a fast and accurate orientation component, the efficient computation of oriented BRIEF and analysis of variance and correlation of oriented BRIEF features.

ORB also uses learning method for de-correlating BRIEF features under rotational invariance, leading to better performance in nearest-neighbor applications. The recognize method for object recognition is Scale invariant feature transform (SIFT), which is popular for its invariance to scaling, rotation and illumination, is computationally complex due to its heavy workload required in local feature extraction and matching operation. Thus computer vision kind of applications demands high performance and low complexity solution and ORB provides better solution to it.

This paper organizes as follows, section II covers the introduction of different algorithms and section III is containing the details of ORB algorithm. Section IV covers the experimentation overview those prove the efficiency of ORB. Section V is the comparison between various adopted methods of object recognition. Section VI gives how FPGA is suited for DSP algorithm implementation than DSP processor. Section VII describes the methodology that can be used to implement the ORB algorithm on FPGA. Section VIII summarizes this review work in terms of conclusion.

2. SIFT and SURF

SIFT

Feature-based image matching is a key task in many computer vision applications, such as object recognition, images stitching, structure-from-motion and 3D stereo reconstruction. These applications require often real-time performance [4].

SIFT can be explained by following stages:
(1) Scale-space peak selection, (2) Keypoint localization,(3) Orientation assignment, (4) Keypoint descriptor. First stage consists of potential interest points which are identified by scanning the image over location and scale. This is implemented efficiently by constructing a Gaussian pyramid and searching for local peaks (termed keypoints) in a series of difference-of-Gaussian (DoG) images. Secong stage includes localization of candidate

keypoints to sub-pixel accuracy and are eliminated if found to be unstable.

The third identifies the dominant orientations for each keypoint based on its local image patch. The assigned orientation(s), scale and location for each keypoint enables SIFT to construct a canonical view for the keypoint that is invariant to similarity transforms.

The final stage builds a local image descriptor for each keypoint, based upon the image gradients in its local neighborhood. The final (keypoint descriptor) stage of the SIFT algorithm builds a representation for each keypoint based on a patch of pixels in its local neighborhood. Note that the patch has been previously centered about the keypoint's location, rotated on the basis of its dominant orientation and scaled to the appropriate size.

The standard keypoint descriptor used by SIFT is created by sampling the magnitudes and orientations of the image gradient in the patch around the keypoint, and building smoothed orientation histograms to capture the important aspects of the patch.

SURF

SURF (Speeded Up Robust Features) [5] presents a novel scale- and rotation-invariant interest point detector and descriptor. It approximates or even outperforms previous schemes with respect to repeatability, distinctiveness, and robustness, yet can be computed and compared much faster.

This is achieved by relying on integral images for image convolutions; by building on the strengths of the leading existing detectors and descriptors (e.g., using a Hessian matrix-based measure for the detector, and a distribution-based descriptor); and by simplifying these methods to the essential. This leads to a combination of novel detection, description, and matching steps.

The detector is based on the Hessian matrix [11, 1], but uses a very basic approximation, just as DoG is a very basic Laplacian-based detector. It relies on integral images to reduce the computation time hence called the 'Fast-Hessian' detector. The descriptor, on the other hand, describes a distribution of Haar-wavelet responses within the interest point neighborhood. Again, integral images are exploited for speed.

3. ORB

Very less literature is present as far as ORB is concerned. This section describes the ORB and how it differs from others.ORB [1] is a very fast binary descriptor based on BRIEF, which is rotation invariant and resistant to noise. It can be demonstrated through experiments how ORB is at two orders of magnitude faster than SIFT, while performing as well in many situations. Authors have tested the efficiency of ORB on several real-world applications, including object detection and patch-tracking on a smart phone.

ORB is made from the well-known FAST keypoint detector and the recently-developed BRIEF descriptor; for this reason we call it ORB (Oriented FAST and Rotated BRIEF). These techniques are provide good performance and have low cost. A fast and accurate orientation component is added to FAST, also efficient computation of oriented BRIEF features, analysis of variance and correlation of oriented BRIEF features and a learning method for de-correlating BRIEF features under rotational invariance, leading to better performance in nearest-neighbor applications.

oFAST: FAST Keypoint Orientation

FAST features are widely used because of their computational properties. However, FAST features do not have an orientation component. oFAST is the efficiently computed orientation added to the FAST.

FAST Detector

FAST takes one parameter, the intensity threshold between the center pixel and those in a circular ring about the center. FAST does not produce a measure of cornerness, and it has large responses along edges. Harris corner measure to order the FAST keypoints is employed. For a target number N of keypoints, it sets the threshold low enough to get more than N keypoints, then orders them according to the Harris measure, and picks the top N points. FAST does not produce multi-scale features. A scale pyramid is employed of the image, and produces FAST features (filtered by Harris) at each level in the pyramid.

ORB uses a simple but effective measure of corner orientation, the *intensity centroid*. The intensity centroid assumes that a corner's intensity is offset from its center, and this vector may be used to impute an orientation. The moments of a patch can be defined as:

$$m_{pq} = \sum_{x,y} x^p y^q \, I(x,y)$$

and with these moments we may find the centroid:

$$C = \left(\frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}} \right)$$

We can construct a vector from the corner's center, O, to the centroid, OC. The orientation of the patch then simply is:

$$\theta = \text{atan2}(m_{01}, m_{10})$$

where atan2 is the quadrant-aware version of arctan
The centroid method is compared with two gradient based measures, BIN and MAX. In both cases, X and Y gradients are calculated on a smoothed image. MAX chooses the largest gradient in the keypoint patch; BIN forms a histogram of gradient directions at 10 degree intervals, and picks the maximum bin. BIN is similar to the SIFT algorithm, although it picks only a single orientation. The variance of the orientation in a simulated dataset is shown in figure. Neither of the gradient measures performs very well, while the centroid gives a uniformly good orientation, even under large image noise. From [1] it can be seen that the intensity centroid (IC) performs best on recovering the orientation of artificially rotated noisy patches, compared to a histogram (BIN) and MAX method.

rBRIEF: Rotation-Aware Brief

The BRIEF descriptor is a bit string description of an image patch constructed from a set of binary intensity tests. Consider a smoothed image patch, p. A binary test τ is defined by:

$$\tau(p; x, y) = \begin{cases} 1 : p(x) < p(y) \\ 0 : p(x) < p(y) \end{cases}$$

where p(x) is the intensity of p at a point x.
The feature is defined as a vector of n binary tests:

$$f_n(p) = \sum_{1 \le i \le n} 2^{i-1} \tau\,(p; x_i, y_i)$$

Steered BRIEF

Matching performance of BRIEF falls off sharply for in-plane rotation of more than a few degrees. A more efficient method is to steer BRIEF according to the orientation of keypoints. For any feature set of n binary tests at location (x_i, y_i), define the 2 × n matrix

$$S = \begin{pmatrix} x_1, \ldots\ldots\ldots x_n \\ y_1, \ldots\ldots\ldots y_n \end{pmatrix}$$

Using the patch orientation θ and the corresponding rotation matrix R_θ, we construct a "steered" version S_θ of S:

$$S_\theta = R_\theta S$$

Now the steered BRIEF operator becomes:

$$g_n(p, \theta) = f_n(p) | (x_i, y_i) \in S_\theta$$

Variance and Correlation

High variance makes a feature more discriminative, since it responds differentially to inputs. Another desirable property is to have the tests uncorrelated, since then each test will contribute to the result. To analyze the correlation and variance of tests in the BRIEF vector, the response to 100k keypoints for BRIEF and steered BRIEF is checked by authors in [1].Both BRIEF and steered BRIEF exhibit high initial eigen values, indicating correlation among the binary tests – essentially all the information is contained in the first 10 or 15 components. Steered BRIEF has significantly lower variance, however, since the eigen values are lower, and thus is not as discriminative.

r-BRIEF

To recover from the loss of variance in steered BRIEF, and to reduce correlation among the binary tests, authors have suggested a learning method for choosing a good subset of binary tests. One possible strategy is to use PCA or some other dimensionality-reduction method, and starting from a large set of binary tests, identifies 256 new features that have high variance and are uncorrelated over a large training set. However, since the new features are composed from a larger number of binary tests, they would be less efficient to compute than steered BRIEF. Instead, authors search among all possible binary tests to find ones that both have high variance (and means close to 0.5), as well as being uncorrelated.

To improve the variance and correlation in steered BRIEF, authors in [1] have suggested one algorithm which is given below. For the algorithm to be implemented, a training set of some 300k points is set up, drawn from the images. Also all possible binary tests drawn from a 31×31 pixel patch are enumerated. Each test is a pair of 5×5 sub-windows of the patch. Note the width of the patch as wp=31 and the width of the test sub-window as wt=5, then there are $N = (wp - wt)^2$ possible sub-windows. By selecting pairs of two from these, so we have N/2 binary tests. Eliminate tests that overlap, so end up with M = 205590 possible tests.

The algorithm is:
1. Run each test against all training patches.
2. Order the tests by their distance from a mean of 0.5, forming the vector T.
3. Greedy search:
(a) Put the first test into the result vector R and remove it from T.
(b) Take the next test from T, and compare it against all tests in R. If its absolute correlation is greater than a threshold, discard it; else add it to R.

(c) Repeat the previous step until there are 256 tests in R. If there are fewer than 256, raise the threshold and try again.
This algorithm is a greedy search for a set of uncorrelated tests with means near 0.5. The result is called *r*BRIEF. rBRIEF has significant improvement in the variance and correlation over steered BRIEF.

4. Case Studies

This section shows the efficiency of ORB over different available object recognition techniques. Authors in [1] evaluate the combination of oFAST and rBRIEF, which is called ORB, using two datasets: images with synthetic in-plane rotation and added Gaussian noise, and a real-world dataset of textured planar images captured from different viewpoints. For each reference image, the oFAST keypoints and rBRIEF features are computed, targeting 500 keypoints per image. For each test image (synthetic rotation or real-world viewpoint change), do the same, then perform brute-force matching to find the best correspondence[1][6].
It can be seen that the standard BRIEF operator falls off dramatically after about 10 degrees.

The synthetic test set with added Gaussian noise of 10

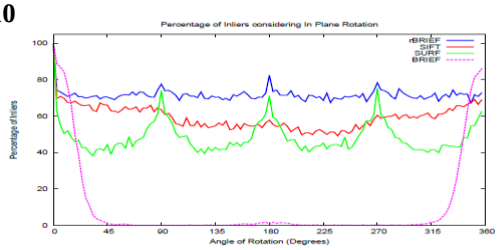

Fig 1 : The synthetic test set with added Gaussian noise of 10 [1]

SIFT outperforms SURF, ORB has the best performance, with over 70% inliers.
The inlier performance vs. noise:

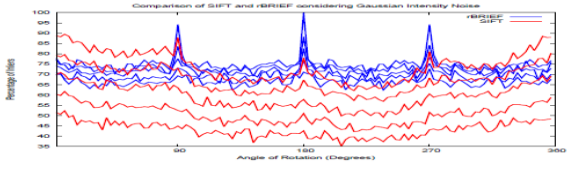

Fig 2 : The inlier performance vs. noise,at different noise levels [1]

5. Comparison of SIFT,SURF and ORB

ORB is rotation invariant and resistant to noise. It is demonstrated through experiments that ORB is at two orders of magnitude faster than SIFT, while performing as well in many situations [1]. The efficiency is tested on several real-world applications, including object detection and patch-tracking on a smart phone. Another advantage of ORB is its very low memory requirement. Its descriptor provides comparable precision/recall results with SURF and SIFT [6]. Following table gives the comparison between SIFT, SURF and ORB.

Table 1: Comparison of SIFT, SURF and ORB

Method	Time per frame	Noise immunity	Rotation	Indoor Data Matching	Outdoor Data Matching
SIFT	Low	Moderate	Moderate	Moderate	Moderate
SURF	Moderate	Low	Low	Moderate	Moderate
ORB	High	High	high	Moderate	High

6. DSP Vs. FPGA

DSP functions are commonly implemented on two types of programmable platforms: digital signal processors and field programmable gate arrays (FPGAs). Digital signal processors are a specialized form of microprocessor, while FPGAs are a form of highly configurable hardware. In the past, the use of digital signal processors was nearly ubiquitous, but with the needs of many applications outstripping the processing capabilities of digital signal processors (measured in millions of instructions per second (MIPS)), the use of FPGAs is growing rapidly. Currently, the primary reason most engineers choose use a FPGA over digital signal processors is driven by the application's MIPS requirements. Thus, the comparison between digital signal processors and FPGAs focuses on MIPS comparison which, while certainly important, is not the only advantage of an FPGA. Equally important, and often overlooked, is the FPGA's inherent advantage in product reliability and maintainability. Nearly all engineering project managers can readily quote the date of their next product software update, or release. At most technology companies, there is usually a long internal list of software bugs or problem reports along with the software releases that will contain the associated

patches, or fixes. It has come to be expected that all software, including DSP code, will contain some level of bugs and that the best one can do is to minimize this. By comparison, FPGA designs tend to be updated much less frequently, and it is generally an unusual event for a manufacturer to issue a field upgrade of a FPGA configuration file [7]. Microprocessor, digital signal processor, and operating system (OS) vendors have attempted to address these problems by creating different levels of protection or isolation between tasks or threads. The operating system, or kernel, is used to manage access to the processor resources, such as allowable execution time, memory, or common peripheral resources. However, there is an inherent conflict between processing efficiency and the level of protection offered by the OS. In digital signal processors (shown in Figure 4), where processing efficiency and deterministic latency are often critical, the result is usually minimal or zero OS isolation between tasks. Each task often requires unrestricted access to many processor resources in order to run efficiently.

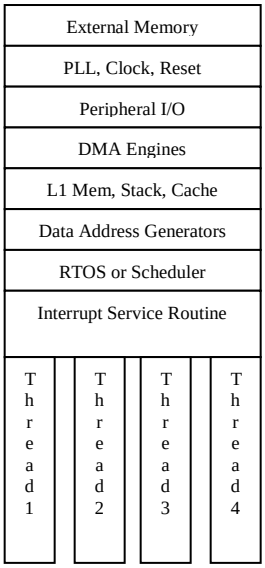

Fig 3: Digital Signal Processor Block Diagram

As products grow in complexity, the lines of code increase, as do the number of processor cores, and an ever-greater percentage of the development effort is devoted to software testing. So how exactly does the FPGA development process improve on this unhappy state of affairs? The complexity of each task is more or less equivalent, no matter whether the design uses digital signal processor or FPGA implementation. Both routes offer the option to use third-party implementations of common signal processing algorithms, interfaces, and protocols. Each also offers the ability to reuse existing intellectual property (IP) on future designs, but that is where the similarity ends, FPGAs offer a more native implementation for most DSP algorithms. Each task is allocated its own resources, and runs independently. It intuitively makes more sense to process each step of a continuously streaming signal processing chain in an assembly line-like process, with dedicated resources for each step.

The FPGA resources assigned can be tailored to the task requirements, which can be broken up along logical partitions. This makes for a well-defined interface between tasks, and largely eliminates unexpected interaction between tasks. Because each task runs continuously, much less memory is required than in the digital signal processor, which must buffer the data and process it in batches.

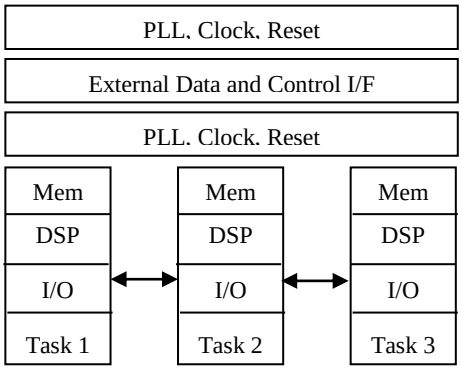

Fig.4: FPGA Block Diagram

As FPGAs distribute memory throughout the device, each task is permanently allocated the dedicated memory it needs. This provides a high degree of isolation between tasks and results in modification of one task being unlikely to cause unexpected behavior in another task. This, in turn, allows developers to easily isolate and fix bugs in a logical and predictable fashion.

7. Methodology

As discussed above, there are two types of technologies available for hardware design. Full custom hardware design also called as Application Specific Integrated Circuits (ASIC) and semi-custom hardware device, which are programmable devices like Digital signal processors (DSPs) and Field Programmable Gate Arrays (FPGA's).

Field Programmable Gate Arrays are reconfigurable devices. Hardware design techniques such as parallelism and pipelining techniques can be developed on a FPGA, which is not possible in dedicated DSP designs. Implementing image processing algorithms on reconfigurable hardware minimizes the time-to-market cost, enables rapid prototyping of complex algorithms and simplifies debugging and verification. Therefore, FPGAs are an ideal choice for implementation of real time image processing algorithms.

The entire implementation of image acquisition, image processing and image retrieval is shown in block diagram [8]. The proposed methodology uses FPGA for the implementation of ORB. The entire ORB algorithm is implemented in different modules.As shown in figure, the architecture includes three different modules those are implemented on FPGA: 1. UART 2.RAM 3.ORBAlgorithmUART is implemented to felicitate data acquisition and communication between PC and FPGA board.

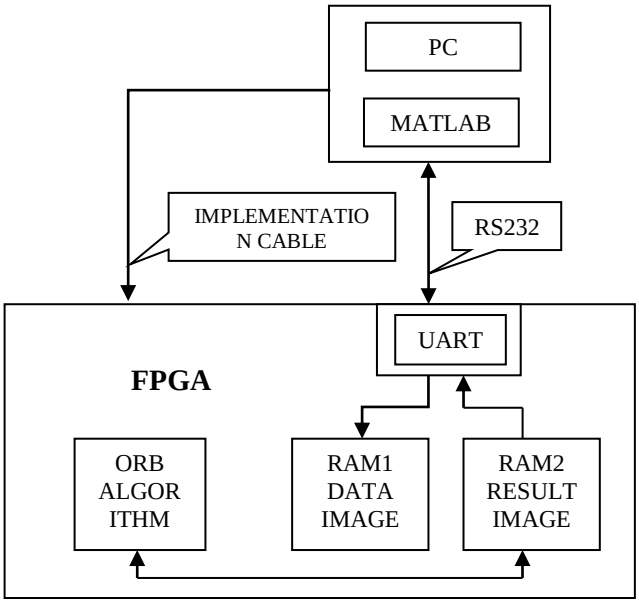

Fig 5: Block Diagram of ORB implementation on FPGA [8]

In order to reduce complexity of data transactions, RAM is implemented on FPGA. Separate RAMs are used for the data image and result image for speeding up the calculations.

8. Conclusion

ORB implementation includes, the addition of a fast and accurate orientation component to FAST, the efficient computation of oriented BRIEF features, analysis of variance and correlation of oriented BRIEF features, a learning method for de-correlating BRIEF features under rotational invariance, leading to better performance in nearest-neighbor applications.

With the addition of new techniques, ORB outperforms SIFT and SURF on the outdoor dataset. It is about the same on the indoor set noted that blob detection key points like SIFT tend to be better on graffiti type images. ORB outperforms SIFT/SURF in nearest-neighbor matching over large databases of images. Implementation of the algorithm on the FPGA than DSP processor increases the speed of execution and provides flexibility due to the pipelining and reconfigurable nature of the FPGA.

References

[1] Ethan Rublee Vincent Rabaud Kurt Konolige Gary Bradski Willow Garage, Menlo Park, "ORB: an efficient alternative to SIFT or SURF", California {erublee}{vrabaud}{konolige}{bradski}@willowgarage.com .

[2] Michael B. Holte, Student Member, IEEE, Cuong Tran, Student Member, IEEE, Mohan M. Trivedi, Fellow, IEEE, and Thomas B. Moeslund, Member, IEEE. "Human Pose Estimation and Activity Recognition From Multi-View Videos: Comparative Explorations of Recent Developments".

[3] Erbert Bay, Tinne Tuytelaars, and Luc Van Gool "SURF: Speeded Up Robust Features".

[4] Yan Ke, Rahul Sukthankar, "PCA-SIFT: A More Distinctive Representation for Local Image Descriptors".

[5] N. Battezzati, S. Colazzo, M. Maffione, L. Senepa, Sky technology, C.so Svizzera, "SURF Algorithm in FPGA: a Novel Architecture for High Demanding Industrial Applications", 185/bis 10149, Torino, ITALY.

[6] Ondrej Miksik,Ondrej Miksik, "Evaluation of Local Detectors and Descriptors for Fast Feature Matching".

[7] FPGA vs. DSP Design Reliability and Maintenance, White Paper.
Ajay Kumar Garg,"Implementation of Image Processing Algorithm on FPGA", by Abdul Manan,Department of Electronics and Communication Engineering, Engineering College, PO Adhyatmic Nagar,Ghaziabad 201009 UP.

Aniruddha Kulkarni has received his B.E. degree from, University of Pune in 2012. He is currently pursuing M.E in VLSI and Embedded Systems from P.C.C.O.E, Pune University.

Jayawant Jagtap has received his B.E. degree from, University of Pune in 2008. He is currently pursuing M.E in VLSI and Embedded Systems from P.C.C.O.E, Pune University and is working in VLSI domain.

Prof. Varsha Harpale is an assistant professor of Electronics and Telecommunication engineering at Pimpri Chinchwad College of Engineering, University of Pune, India. She received her B.E and M.E. degrees from University of Pune. She is persuing her PhD from the university of Pune and her research work includes video processing using different technologies.