Journal of Mobile Multimedia, Vol. 12, No. 3&4 (2017) 291–302

ANALYSIS AND EVALUATION OF FEATURE DETECTION AND TRACKING TECHNIQUES USING OPENCV WITH FOCUS ON MARKERLESS AUGMENTED REALITY APPLICATIONS

GUSTAVO MAGALHÃES MOURA RODRIGO LUIS DE SOUZA DA SILVA
Computer Science Department, Federal University of Juiz de Fora
Juiz de Fora, Minas Gerais, 36036-900, Brazil
gmmoura@ice.ufjf.br rodrigoluis@ice.ufjf.br

Augmented Reality (AR) is a technology able to extend human interactions with the real world. One field of study in AR is the use of real objects as markers. To perform this task, feature recognition of the real world by computer systems must be performed. This work consists in the analysis and evaluation of several algorithms available in OpenCV library that allow the detection of pre-established patterns in images and videos. The main contribution of this work is to present the most appropriate combination of algorithms to help the development of markerless AR applications.

Keywords: Augmented Reality, OpenCV, feature detector

1 Introduction

Augmented Reality (AR) is a technology that allows the modification of a real-world view through the overlay of virtual images, generated computationally, using a capture device. In [1], AR is defined as a system that complements the real-world with virtual objects generated by computers that appear to coexist in the same space as the real world. The generation of these virtual objects can be made through sensing and overlapping patterns or images with georeferenced points.

Although augmented reality research dates back to the early 1960s through the work of Sutherland [2], its popularity has not yet occurred, compared with the recent growth of virtual reality, it is mainly restricted to research environments and industrial applications. With the advent of relatively inexpensive equipment, hardware that support the requirements of AR applications, for example smartphones, and libraries that support the development of AR applications, the range of this technology is increasing. Among the available libraries that provide algorithms for the development of AR applications there is OpenCV.

OpenCV (Open Source Computer Vision) is a free and open library created in 1998 by Intel, currently held by Itseez, which provides tools for developing applications in Computer Vision for academic and commercial use. It has interfaces for C++, C, Python, Java and MATLAB and support for development on Linux, Windows, Mac OS, iOS and Android. OpenCV has over 2500 algorithms that support basic and advanced image processing, computational geometry, detector and feature descriptors, object tracking, optical flow, object detection, camera calibration, etc.

OpenCV has wide applicability, from health [3] to traffic control [4] and it was chosen for this work to be free, robust, support multiple programming languages and have user-friendly

interface to access their algorithms.

Pattern detection in an image basically comprises three stages: feature detection, description and matching. There are specific algorithms for each step and algorithms comprising the first two. This work will present the combination of three types of algorithms available in the OpenCV library for detecting a predetermined pattern in an image (Figure 1). The main contribution of this work is to present the most appropriate combination of algorithms for AR applications.

Fig. 1. Example of matching a pattern in an image (left) on a set of images where the pattern is inserted (right).

The remainder of this paper is organized as follows: Section 2 describes the related works; Section 3 introduces the concepts and algorithms used in this work; the description of the methodology was carried out in Section 4; experiments are presented in Section 5; and the conclusion is found in Section 6.

2 Related Works

Among the three types of algorithms presented in this paper, detectors and descriptors are the most widely studied because of the dependency on what information is relevant in the image and how they are encoded, respectively, and they demand the highest percentage of processing time, as will be presented in the following sections.

Due to the large number of algorithms responsible for detection and description available in the literature, their versatility and the transformations that an image may suffer, comparison articles can evaluate different aspects with different emphases, as in [5]. This paper compares two of the most used algorithms, namely detectors and descriptors, focusing on matching different views of the same scene, assessing the number of interest points and processing time. A comparison between five algorithms for detection and description in order to study the correspondence of a standard object in an image was presented in [6]. In this work the authors did not set an unique matching algorithm, because the types of descriptors used were different.

Other studies have brought comparisons with a large number of algorithms and changes

in test images such as image quality, scale, rotation and perspective transformations. In [7] the best algorithm, or combination of each evaluated transformation was presented. The work presented in [8] used a greater number of changes in test images, lighting, viewpoint changes, rotation, blurring, scale, JPEG compression and exposure time, and used combinations of detectors and descriptors to determine the best combination for each evaluated transformation.

The papers [9] and [10] compare the two most widely used algorithms for detection and description and proposed new approaches to generate new algorithms.

The work that is closest to this one was presented in [11]. This work compare the detection and description algorithms and two matching algorithms aiming to get the best fit for visual recognition of pictures in smartphone.

The main contribution of this work in comparison with the articles presented in this section is the inclusion of the matching algorithms presented in OpenCV regarding performance reviews and the evaluation of combinations to detect a pattern previously established in images and videos with focus on the performance requirements for AR applications.

3 Basic Concepts

This section describes the basic concepts presented in this paper.

3.1 Feature Detectors

In [12], Suarez et al defined that feature detectors are algorithms that extract points of an image that can characterize its contents robustly. These points (also called local features or interest points) can be classified into edges, corners or blobs (regions). The regions can be considered as areas where certain properties are approximately constant. The detection algorithms are specialized in identifying one or more of the mentioned characteristics. Table 1 shows the types of characteristics that each detection algorithm considers.

Table 1. Types of features considered by each detection algorithm

Detector	**Feature Type**		
	Edge	**Corner**	**Blob**
BRISK		✓	✓
Dense	Detects the type of feature that is regularly distributed in the image.		
FAST		✓	✓
GFTT	✓	✓	
HARRIS	✓	✓	
MSER			✓
ORB		✓	✓
SIFT		✓	✓
SimpleBlob			✓
STAR	✓	✓	✓
SURF		✓	✓

Proposed by [13], **FAST** (*Features from Accelerated Segment Test*) detector is very fast in detecting corners. However, it is not selective detecting many points of no great interest.

BRISK (*Binary Robust Invariant Scalable Keypoints*) detector proposed by [14] was inspired by FAST detector. It was presented as an alternative to SIFT and SURF with the purpose to maintain their strengths, but with better performance.

Dense algorithm is a detector that picks the interest points on the nodes of a regular grid superimposed on the image excluding the points of the grid limits. Because of this feature, the selected interest points may not be relevant.

GFTT (*Good Features To Track*) method proposed by [15], tries to detect corners and edges that stand out in the images, since they are less affected by rotation or image scale.

HARRIS detector combines the GFTT method with the detection method proposed by [16]. This detector handles corners and edges tolerating rotations, but is not suitable for scale changes.

MSER (*Maximally Stable Extremal Regions*), proposed by [17], is a method able to detect regions, being tolerant to rotations, change of scale, perspective and lighting changes.

Created to be an alternative to the SIFT and SURF, the **ORB** (*Oriented FAST and Rotated BRIEF*) algorithm was proposed by [9] and its detector was set from FAST adding an orientation component. The authors claimed that their method is fast and versatile when compared to similar approaches.

Presented by [18], the **SIFT** (*Scale Invariant Feature Transform*) method is one of the most popular detectors in literature. The method processes the image pixel by pixel with a high computational cost, but it has the advantage of finding interest points invariant to scale and rotation. This method is patented and can not be used for commercial purposes without authorization.

SimpleBlob algorithm uses a simple method to detect regions, as the name suggests. The algorithm converts the original image into two or more colors, generating images in grayscale. After that, the grayscale values are binarized. The common areas in both images are then filtered considering various parameters. This method is tolerant to rotations.

STAR algorithm is an implementation of OpenCV derived from the CenSurE (*Center Surround Extremas*) detector, proposed by [19]. The CenSurE uses polygons as squares, hexagons and octagons to delimit the search for regions as a less computationally expensive alternative. STAR uses as a delimiter two overlapping squares, one of them rotated 45°. The obtained figure is called regular octagram {8}2, which is an eight-pointed star.

SURF (*Speeded Up Robust Features*) is a method proposed by [20], having a robust feature detector and it is tolerant to scale and rotation. It is similar to the SIFT method to obtain interest points and the most important difference is the method used to determine the valid interest points. This method also has patent and can not be used for commercial purposes.

3.2 Descriptors

Descriptor algorithms transform the information obtained from interest points by the detector algorithms in information that is invariant to differences in lighting and small perspective deformations for use by matching algorithms. Some detection algorithms also has components of description. A brief explanation of the descriptors used in this work will be presented next.

BRIEF (*Binary Robust Independent Elementary Features*) descriptor was the first binary descriptor proposed by [21]. This descriptor does not have a orientation component nor a interest points sampling pattern obtained by the detector.

Proposed by [14], the **BRISK** algorithm is a binary descriptor using an orientation component obtained by analyzing the neighboring pixels to the interest points.

FREAK (*Fast Retina Keypoint*) method is a descriptor inspired by human visual system

and proposed by [22]. This binary descriptor uses a small amount of memory and has a fast processing.

Also proposed by [9], **ORB** descriptor was based on the BRIEF descriptor, but has an efficient calculation of the interest points orientation and a variance and correlation analysis of interest points to deliver best results.

SIFT descriptor, proposed by [18], uses gradient orientation histograms in the region around the interest points depending on the location of these points in the image. It is a robust descriptor invariant to scale, rotation and other transformations on the images. Due to the high amount of information stored for each point of interest, its computational cost is high.

Also proposed by [20], **SURF** descriptor uses the dominant orientation of the square region surrounding the interest point to determine the orientation of this point.

As shown, there are methods that are only detectors or descriptors and other methods that have detection and description components. Table 2 summarizes the components of the presented methods.

Table 2. Summary of the components of detectors and descriptors evaluated

Method	Detector	Descriptor
BRISK	✓	✓
BRIEF		✓
Dense	✓	
FAST	✓	
FREAK		✓
GFTT	✓	
HARRIS	✓	
MSER	✓	
ORB	✓	✓
SIFT	✓	✓
SimpleBlob	✓	
STAR	✓	
SURF	✓	✓

3.3 Matchers

Matching Algorithms (or *Matchers*) are methods that determine which characteristics represented in the descriptors of two images are similar according to their criteria. Chances of finding a pattern in an image increases with the number of similar features found.

Brute force matchers available in OpenCV are simple. Each feature of the first descriptor is compared with all features of the second descriptor according to a distance metric with the closest pair returned. A minimum distance value determines whether the pair will be considered relevant or not.

A brief description of the matching algorithms used is presented next according to [12]. In the following equations, V_1 and V_2 are the feature vectors of the two images, M the size of the vectors and $v_1[i]$ and $v_2[i]$ are the ith element of the respective vector.

The **BruteForce-L1** algorithm uses the L1 metric distance, also known as *Manhattan* or *City Block*, to determine the distance between floating point descriptors as shown in Eq. 1.

$$d(V_1, V_2) = \sum_{i=1}^{M} |v_1[i] - v_2[i]| \tag{1}$$

BruteForce is used by floating points descriptors and the considered distance is L2, also known as Euclidean Distance. This method requires more processing power than BruteForce-L1 since it is a quadratic function as shown in Eq. 2.

$$d\left(V_1, V_2\right) = \sqrt{\sum_{i=0}^{M} \left(v_1\left[i\right] - v_2\left[i\right]\right)^2} \tag{2}$$

The **BruteForce-Hamming** implementation is used by binary descriptors. Its equation is the sum of the results of a bit-by-bit bitwise XOR operation between description vectors of two images, according to Eq. 3.

$$d\left(V_1, V_2\right) = \sum_{i=1}^{M} v_1\left[i\right] \otimes v_2\left[i\right] \tag{3}$$

BruteForce-Hamming(2) is also used by binary descriptors and uses two bits rather than one bit in XOR operations compared to the previous algorithm.

Proposed by [23], the **FlannBased** algorithm is used with floating point descriptors. It uses a framework for a preprocessing stage, usually faster than brute force algorithms at the cost of greater memory utilization.

4 Methodology

An AR application should recognize a pattern in a scene, and then use it as a reference in the registration of a virtual object. Patterns can be 2D or 3D, but usually 2D patterns (as images) are the most used. These images should preferably be obtained frontally and under favorable light conditions for a proper capture of the features that compose its pattern. When executing the search for this pattern in a second image or a video, its possible to occur geometric distortions such as rotation and scale that make it difficult to detect. Besides, a sequence of images captured in a scene may suffer variation in brightness, contrast, focus, among other visual characteristics that can prevent the identification of a particular marker when moving the camera from its original position. Thus, it is necessary to use a systematic method to capture images in order to prevent the disturbance caused by environmental conditions when analyzing the algorithms described in this paper.

In order to create a sequence of images with more controlled features, we used a graphic design program to build the standard image and the test images, varying only the transformation parameters systematically. Our base pattern (standard image) was a high resolution image obtained from the internet. Test images were constructed using the standard image in the center, with dimensions that correspond to approximately 25% of the test image area. The background was composed by other high-resolution images of the same size positioned at various angles, to obtain a test image with heterogeneous background. Figure 2 exemplifies the construction of the test images.

For the construction of the test videos, we used a video editing program and the same high-resolution images used to build the test images. For geometric transformations, the angle variation rate was 1 degree per frame and the frame rate was 24 fps.

To analyze the aforementioned algorithms, the parameters used for comparison were as follows:

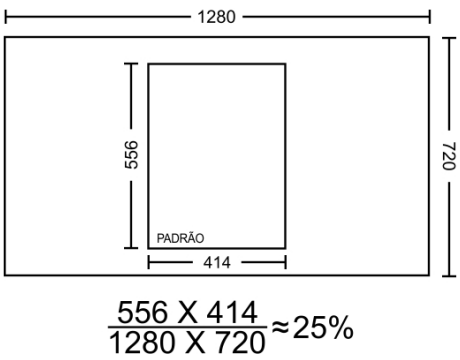

Fig. 2. Example of the construction of test images. Dimension is in pixels.

- **Time**: time required for the processing algorithm or set of algorithms completing their purpose;
- **Accuracy**: number of correctly identified images divided by the total number of test images;
- **Frame rate**: number of frames processed per second (*fps*) by the set of algorithms, exclusively for the analysis of test videos.

5 Experiments

The experiments described here were performed on an Asus notebook model S400CA, Intel® Core™ i5 3317U 1.70 GHz processor, 4GB RAM DDR3 1600 MHz and an Intel® HD Graphics 4000 graphics board.

Algorithms presented in Section 3 were implemented using their respective OpenCV versions. However, for ORB, GFTT and HARRIS detectors, it was necessary to specify the maximum number of points to be detected. Thus, to match all detectors to their maximum number of computable interest points, a sufficiently high value of interest point was specified for these three detectors.

The number of interest points and the execution time of each detector obtained for the standard image and the test image without any transformation with dimensions of 1280 x 720 are listed in Table 3. Dense, FAST and ORB detectors stands out for their large number of interest points detected, and the first two also emphasize the low execution times.

As the initial goal of most Augmented Reality applications is the detection of a pattern in an image, all possible combinations of the algorithms previously described (a total of 330 combinations) was computed, to detect a standard image on a set of test images modified by geometric transformations (mainly rotations on Z and X axis). Axes orientation are illustrated in Figure 3.

Table 4 shows the accuracy (in percentage) obtained for all combinations of algorithms when detecting the base image on the test image set. Some combinations had an error during the tests because some methods were incompatible. These combinations were excluded from the table. The combinations that presented best accuracies considering both geometric transformations are highlighted in Table 4.

Table 3. Detectors Performance

Detector	Pattern		Scene	
	Interest Points	**Time (ms)**	**Interest Points**	**Time (ms)**
BRISK	944	35,29	2518	102,42
Dense	6417	0,13	25680	0,50
FAST	6336	5,75	15415	15,45
GFTT	2972	19,89	6613	77,19
HARRIS	1431	18,68	2612	73,37
MSER	286	238,06	793	818,78
ORB	10885	37,07	34013	112,78
SIFT	1546	195,85	4513	735,89
SimpleBlob	11	90,18	37	234,85
STAR	481	14,04	1933	80,66
SURF	2243	317,95	6809	1120,66

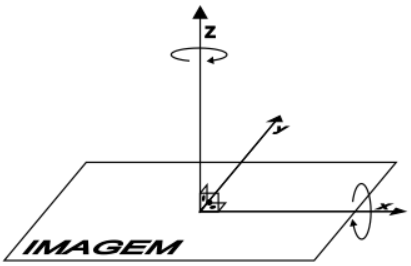

Fig. 3. Orientation of rotation axes of geometric transformations.

For the remainder of this work, we will denote the trio detector, descriptor and matcher algorithms using the notation: detector+descriptor+matcher to simplify the representation.

Table 5 shows the results of the 34 best matches considering test images with dimensions 1280 x 720 images, 540 x 960 and 480 x 360 pixels. For each combination, the percentage accuracy and the average time (in seconds) of a correct match is presented, separated by rotation axis. Combinations with higher accuracy associated with a lower processing time were obtained with the SIFT detector, and the best overall combination for accuracy was SIFT+SIFT+BruteForce-L1. The combinations SIFT+BRISK+BruteForce-Hamming and SIFT+BRISK+BruteForce-Hamming(2) also gave satisfactory results with lower processing time for all test images. Finally, the combinations STAR+SURF+BruteForce, STAR+SURF +FlannBased and STAR+SURF+BruteForce-L1 gave satisfactory accuracy results with the lowest processing times considering all combinations, except with test images with lower dimensions.

Table 6 shows the accuracy and frame rate results for the 18 best combinations when detecting an image pattern in videos with 960 x 540 videos and 480 x 360 pixels dimension. The fastest combinations were SIFT+BRISK+BruteForce-Hamming(2) and SIFT+ BRISK+BruteForce-Hamming, however their accuracy results for tests in videos with 960 x 540 pixels were not satisfactory.

The combinations SIFT+SIFT+BruteForce, SIFT+SIFT+FlannBased and SIFT+SIFT+ BruteForce-L1 showed best accuracy results, however, their best results of frame rate was

Table 4. Accuracy of test images with dimensions of 1280 x 720 pixels separated by rotation axis. The combinations that presented best accuracies considering both geometric transformations are highlighted.

	Z Axis											X Axis										
	BK	**DE**	**FA**	**GF**	**HA**	**MS**	**OR**	**SI**	**SB**	**ST**	**SU**	**BK**	**DE**	**FA**	**GF**	**HA**	**MS**	**OR**	**SI**	**SB**	**ST**	**SU**
BF	0,0	4,2	0,0	4,2	8,3	4,2	4,2	4,2	0,0	8,3	8,3	33,3	16,7	0,0	16,7	33,3	33,3	0,0	16,7	16,7	33,3	33,3
BK	0,0	4,2	62,5	54,2	37,5	16,7	41,7	91,7	0,0	50,0	70,8	0,0	66,7	33,3	16,7	0,0	16,7	33,3	50,0	0,0	50,0	66,7
FR	0,0	0,0	0,0	0,0	0,0	0,0	0,0	-	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	-	0,0	0,0	0,0
OR	0,0	4,2	41,7	33,3	20,8	0,0	66,7	-	0,0	8,3	79,2	0,0	16,7	16,7	16,7	16,7	0,0	33,3	-	0,0	16,7	16,7
SI	**100,0**	-	12,5	12,5	12,5	12,5	**100,0**	**100,0**	4,2	12,5	**100,0**	**66,7**	-	83,3	83,3	83,3	83,3	**50,0**	**83,3**	33,3	83,3	**66,7**
SU	**95,8**	-	**100,0**	25,0	29,2	**100,0**	**100,0**	**95,8**	16,7	**100,0**	**100,0**	**66,7**	-	**50,0**	0,0	0,0	**50,0**	**66,7**	**50,0**	0,0	**50,0**	**66,7**

(a) BruteForce

	BK	**DE**	**FA**	**GF**	**HA**	**MS**	**OR**	**SI**	**SB**	**ST**	**SU**	**BK**	**DE**	**FA**	**GF**	**HA**	**MS**	**OR**	**SI**	**SB**	**ST**	**SU**
BF	8,3	4,2	4,2	8,3	8,3	8,3	4,2	4,2	0,0	8,3	8,3	33,3	16,7	0,0	16,7	33,3	16,7	0,0	16,7	16,7	33,3	33,3
BK	4,2	8,3	87,5	91,7	91,7	33,3	41,7	66,7	0,0	62,5	50,0	0,0	66,7	50,0	50,0	66,7	33,3	33,3	66,7	0,0	50,0	50,0
FR	0,0	0,0	0,0	-	0,0	0,0	0,0	-	-	0,0	0,0	0,0	0,0	0,0	-	0,0	0,0	0,0	-	-	0,0	0,0
OR	0,0	0,0	45,8	37,5	50,0	8,3	45,8	-	0,0	45,8	66,7	0,0	16,7	16,7	16,7	33,3	16,7	66,7	-	0,0	16,7	16,7
SI	**100,0**	-	12,5	12,5	12,5	12,5	**100,0**	**95,8**	8,3	20,8	**100,0**	**66,7**	-	83,3	83,3	83,3	66,7	**66,7**	**83,3**	50,0	83,3	**66,7**
SU	**95,8**	-	**100,0**	62,5	62,5	**95,8**	**100,0**	91,7	16,7	**100,0**	**100,0**	**66,7**	-	**66,7**	0,0	0,0	**66,7**	**66,7**	50,0	0,0	**66,7**	**66,7**

(b) BruteForce-L1

	BK	**DE**	**FA**	**GF**	**HA**	**MS**	**OR**	**SI**	**SB**	**ST**	**SU**	**BK**	**DE**	**FA**	**GF**	**HA**	**MS**	**OR**	**SI**	**SB**	**ST**	**SU**
BF	8,3	4,2	8,3	8,3	8,3	8,3	8,3	8,3	0,0	8,3	8,3	83,3	50,0	33,3	50,0	50,0	83,3	50,0	33,3	50,0	33,3	50,0
BK	4,2	12,5	100,0	91,7	87,5	25,0	75,0	**95,8**	0,0	70,8	62,5	0,0	66,7	33,3	50,0	16,7	33,3	50,0	**66,7**	0,0	50,0	66,7
FR	0,0	-	0,0	0,0	0,0	0,0	0,0	-	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	-	0,0	0,0	0,0
OR	0,0	4,2	79,2	91,7	79,2	8,3	**95,8**	-	4,2	75,0	83,3	0,0	33,3	33,3	50,0	50,0	16,7	**50,0**	-	0,0	33,3	50,0

(c) BruteForce-Hamming

	BK	**DE**	**FA**	**GF**	**HA**	**MS**	**OR**	**SI**	**SB**	**ST**	**SU**	**BK**	**DE**	**FA**	**GF**	**HA**	**MS**	**OR**	**SI**	**SB**	**ST**	**SU**
BF	8,3	4,2	8,3	8,3	8,3	8,3	8,3	8,3	0,0	8,3	8,3	83,3	33,3	33,3	33,3	50,0	66,7	33,3	33,3	50,0	33,3	33,3
BK	4,2	12,5	100,0	91,7	79,2	20,8	75,0	**95,8**	4,2	62,5	62,5	0,0	66,7	33,3	50,0	16,7	16,7	50,0	**66,7**	0,0	50,0	66,7
FR	0,0	-	0,0	0,0	0,0	0,0	0,0	-	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	-	0,0	0,0	0,0
OR	0,0	12,5	79,2	91,7	75,0	25,0	**95,8**	-	4,2	70,8	83,3	0,0	33,3	33,3	50,0	50,0	16,7	**50,0**	-	0,0	33,3	50,0

(d) BruteForce-Hamming(2)

	BK	**DE**	**FA**	**GF**	**HA**	**MS**	**OR**	**SI**	**SB**	**ST**	**SU**	**BK**	**DE**	**FA**	**GF**	**HA**	**MS**	**OR**	**SI**	**SB**	**ST**	**SU**
SI	**100,0**	-	12,5	12,5	12,5	12,5	**100,0**	**100,0**	4,2	12,5	**100,0**	**66,7**	-	83,3	83,3	83,3	66,7	**50,0**	**83,3**	33,3	83,3	**66,7**
SU	**95,8**	-	**100,0**	29,2	37,5	**100,0**	**100,0**	**95,8**	16,7	**100,0**	**100,0**	**66,7**	-	**50,0**	0,0	0,0	**50,0**	**66,7**	**50,0**	0,0	**50,0**	**50,0**

(e) FlannBased

Lines are the descriptors and columns the detectors: *BF* - BRIEF, *BK* - BRISK, *DE* - Dense, *FA* - FAST, *FR* - FREAK, *GF* - GFTT, *HA* - HARRIS, *OR* - ORB, *SB* - SimpleBlob, *SI* - SIFT, *ST* - STAR, *SU* - SURF.

between 2.2 and 1.6 slower when compared with best frame rate combinations.

Considering the 18 combinations listed in Table 6, 16 used at least one of the SIFT or SURF components, which are patented methods, therefore depending on the authorization of the patent for commercial use. The two combinations using only free methods are ORB+ORB+BruteForce-Hamming and ORB+ORB+BruteForce-Hamming(2) which obtained intermediate results of accuracy and frame rate.

ORB is a free method with 8 configurable parameters. This method obtained superior processing time in previous works (as in [8] and [6]) when compared to other methods. To achieve better results, additional tests were performed varying the maximum amount of points of interest that the ORB detector can find. Tables 7 and 8 contain the lower limits of interest points that achieve best results with images and videos, respectively, with 480 x 360 pixels.

Comparing the results of Tables 5 and 7, using the combinations ORB+ORB+BruteForce-Hamming and ORB+ORB+BruteForce-Hamming(2) the accuracy results and average time of success were higher when the number of interest points was limited to a low value when compared with the results of these combinations with a high number of interest points. The average time for the ORB+ORB+BruteForce-Hamming combination was superior when compared to the best results of Table 5.

Considering the combinations ORB+ORB+BruteForce-Hamming and ORB+ORB+BruteForce-Hamming(2) in Tables 6 and 8, changing the number of interest points did not change significantly the accuracy but increased the frame rate when the number of interest points was reduced. This behavior was expected, because reducing the number of interest points

Table 5. Accuracy results and average frame rate to detect a pattern in test images.

Detector	Descritor	Matcher	1280 x 720				960 x 540				480 x 360			
			Z Axis		X Axis		Z Axis		X Axis		Z Axis		X Axis	
			A	T	A	T	A	T	A	T	A	T	A	T
BRISK	SIFT	BruteForce	100,0	1,98	66,7	1,91	100,0	1,39	50,0	1,39	0,0	-	0,0	-
BRISK	SIFT	BruteForce-L1	100,0	1,95	66,7	1,88	100,0	1,37	66,7	1,37	12,5	0,55	0,0	-
BRISK	SIFT	FlannBased	100,0	1,89	50,0	1,85	100,0	1,36	66,7	1,36	0,0	-	0,0	-
BRISK	SURF	BruteForce	95,8	0,75	66,7	0,71	100,0	0,47	66,7	0,49	58,3	0,15	50,0	0,18
BRISK	SURF	BruteForce-L1	95,8	0,73	66,7	0,69	100,0	0,46	50,0	0,47	41,7	0,15	33,3	0,16
BRISK	SURF	FlannBased	95,8	0,7	66,7	0,66	100,0	0,46	50,0	0,47	58,3	0,16	33,3	0,17
FAST	SURF	BruteForce	100,0	6,04	50,0	6,96	100,0	2,93	66,7	2,92	95,8	0,63	66,7	0,61
FAST	SURF	BruteForce-L1	100,0	5,37	66,7	6,05	100,0	2,63	66,7	2,63	100,0	0,59	66,7	0,56
FAST	SURF	FlannBased	100,0	1,85	50,0	2,13	100,0	1,19	66,7	1,21	95,8	0,45	66,7	0,43
MSER	SURF	BruteForce	100,0	1,13	50,0	1,11	95,8	0,62	50,0	0,65	12,5	0,17	0,0	-
MSER	SURF	BruteForce-L1	95,8	1,13	66,7	1,1	91,7	0,62	66,7	0,63	8,3	0,16	0,0	-
MSER	SURF	FlannBased	100,0	1,13	50,0	1,12	95,8	0,63	50,0	0,66	12,5	0,17	0,0	-
ORB	ORB	BruteForce-Hamming	83,3	4,69	50,0	4,56	83,3	3,46	66,7	3,41	87,5	0,47	66,7	0,48
ORB	ORB	BruteForce-Hamming(2)	83,3	5,32	66,7	5,31	83,3	3,96	50,0	3,88	87,5	0,53	50,0	0,55
ORB	SIFT	BruteForce	100,0	22,14	33,3	22,06	70,8	19,81	33,3	19,85	0,0	-	0,0	-
ORB	SIFT	BruteForce-L1	95,8	21,05	50,0	21,09	79,2	19,08	16,7	19,05	0,0	-	0,0	-
ORB	SIFT	FlannBased	100,0	15,12	50,0	15,13	70,8	14,71	33,3	14,71	0,0	-	0,0	-
ORB	SURF	BruteForce	100,0	15,65	66,7	15,56	95,8	14,3	66,7	14,33	100,0	5,06	50,0	5,28
ORB	SURF	BruteForce-L1	100,0	15	66,7	14,95	100,0	13,84	66,7	13,86	100,0	5	50,0	5,22
ORB	SURF	FlannBased	100,0	11,72	66,7	11,66	95,8	11,43	66,7	11,49	100,0	4,8	50,0	5,01
SIFT	BRISK	BruteForce-Hamming	95,8	1,42	50,0	1,49	91,7	0,77	50,0	0,77	100,0	0,17	66,7	0,17
SIFT	BRISK	BruteForce-Hamming(2)	95,8	1,5	50,0	1,6	91,7	0,82	50,0	0,82	100,0	0,17	66,7	0,17
SIFT	SIFT	BruteForce	100,0	2,35	83,3	2,47	100,0	1,37	83,3	1,37	100,0	0,36	66,7	0,36
SIFT	SIFT	BruteForce-L1	95,8	2,26	83,3	2,38	100,0	1,33	83,3	1,32	100,0	0,36	83,3	0,36
SIFT	SIFT	FlannBased	100,0	1,93	66,7	2,04	100,0	1,18	83,3	1,19	100,0	0,36	66,7	0,37
SIFT	SURF	BruteForce	95,8	1,85	50,0	2,37	87,5	1,22	66,7	1,29	83,3	0,39	66,7	0,39
SIFT	SURF	FlannBased	95,8	1,62	50,0	2,05	83,3	1,11	66,7	1,18	83,3	0,39	50,0	0,38
STAR	SURF	BruteForce	100,0	0,43	50,0	0,43	100,0	0,25	50,0	0,26	0,0	-	0,0	-
STAR	SURF	BruteForce-L1	100,0	0,43	66,7	0,44	95,8	0,25	66,7	0,28	0,0	-	0,0	-
STAR	SURF	FlannBased	100,0	0,43	50,0	0,43	100,0	0,26	50,0	0,27	0,0	-	0,0	-
SURF	SIFT	BruteForce	100,0	13,59	66,7	12,7	100,0	7,8	66,7	7,48	8,3	2,06	16,7	2,03
SURF	SIFT	BruteForce-L1	100,0	13,4	66,7	12,5	100,0	7,74	66,7	7,39	25,0	1,99	50,0	2,04
SURF	SIFT	FlannBased	100,0	12,57	66,7	11,75	100,0	7,5	66,7	7,2	8,3	2,06	33,3	2,01
SURF	SURF	BruteForce	100,0	4,77	66,7	4,58	95,8	2,58	83,3	2,42	87,5	0,58	66,7	0,58
SURF	SURF	BruteForce-L1	100,0	4,65	66,7	4,49	100,0	2,54	66,7	2,51	95,8	0,58	66,7	0,58
SURF	SURF	FlannBased	100,0	4,19	66,7	4,06	95,8	2,41	83,3	2,28	87,5	0,58	50,0	0,61

Where *A* - Accuracy (%), *T* - Average Time (s).

Table 6. Accuracy results and average frame rate to detect a pattern in videos

Detector	Descritor	Matcher	960 x 540				480 x 360			
			Z Axis		X Axis		Z Axis		X Axis	
			A	F	A	F	A	F	A	F
FAST	SURF	BruteForce	97,8	0,34	54,4	0,41	99,4	1,61	58,9	1,76
FAST	SURF	BruteForce-L1	98,6	0,38	57,8	0,45	100,0	1,72	60,0	1,87
FAST	SURF	FlannBased	97,8	0,85	54,4	0,95	99,4	2,29	60,0	2,36
ORB	ORB	BruteForce-Hamming	90,6	0,15	52,2	0,19	95,6	2,31	46,7	2,23
ORB	ORB	BruteForce-Hamming(2)	90,3	0,14	51,1	0,17	96,1	2,05	44,4	2,01
ORB	SURF	BruteForce	99,7	0,05	61,1	0,05	96,1	0,22	41,1	0,22
ORB	SURF	BruteForce-L1	100,0	0,05	62,2	0,05	96,7	0,22	37,8	0,22
ORB	SURF	FlannBased	99,7	0,06	55,6	0,07	96,1	0,23	43,3	0,23
SIFT	BRISK	BruteForce-Hamming	90,6	1,59	58,9	1,51	98,6	6,91	60,0	5,43
SIFT	BRISK	BruteForce-Hamming(2)	91,4	1,50	55,6	1,49	98,3	6,70	60,0	5,28
SIFT	SIFT	BruteForce	100,0	0,80	67,8	0,87	100,0	3,13	63,3	3,02
SIFT	SIFT	BruteForce-L1	99,7	0,83	66,7	0,90	99,7	3,20	67,8	3,09
SIFT	SIFT	FlannBased	100,0	0,92	66,7	0,97	100,0	3,11	65,6	2,97
SIFT	SURF	BruteForce	89,4	0,95	50,0	1,08	82,2	3,07	52,2	3,30
SIFT	SURF	FlannBased	88,9	1,04	50,0	1,16	81,7	3,03	50,0	3,26
SURF	SURF	BruteForce	100,0	0,43	60,0	0,47	91,4	1,94	53,3	1,85
SURF	SURF	BruteForce-L1	100,0	0,43	56,7	0,48	91,4	1,95	52,2	1,86
SURF	SURF	FlannBased	100,0	0,46	60,0	0,50	91,4	1,91	50,0	1,83

Where *A* - Accuracy (%), *F* - *Frame rate* (fps).

Table 7. Combination results of the ORB method in images

Combination	RA	IPL	A	T
ORB+ORB+	Z	1990	95,8	0,14
BruteForce-Hamming	X	910	66,7	0,09
ORB+ORB+	Z	2400	95,8	0,19
BruteForce-Hamming(2)	X	1840	66,7	0,18

Where *RA* - Rotation Axis, *IPL* - Interest Points Limit, *A* - Accuracy (%), *T* - Average Time (s).

Table 8. Combination results of the ORB method in videos

Combination	RA	IPL	A	T
ORB+ORB+	Z	3640	95,0	4,18
BruteForce-Hamming	X	4500	51,1	3,05
ORB+ORB+	Z	3640	94,7	4,26
BruteForce-Hamming(2)	X	4260	51,1	2,84

Where *RA* - Rotation Axis, *IPL* - Interest Points Limit, *A* - Accuracy (%), *T* - Average Time (s).

speed up the detection.

6 Conclusions

The main contribution of this work was to provide a comparison and analysis of combinations of the algorithms presented in OpenCV library that allow the detection of pre-established patterns in images and videos and determine the best combinations for Augmented Reality systems. All combinations of detectors, descriptors and matching algorithms available in the OpenCV library were compared to detect a pattern in images and video with application of rotations and changes in scale.

After the tests, the results demonstrated that, without a previous setting, combinations of the SIFT detector, BRISK descriptor and binary matching algorithms were superior in the detection of a pattern in images and videos, especially when using low resolution.

The ORB method with a fine parameter adjustment was a good alternative to patented methods with satisfactory results, especially when used to detect image patterns.

The average frame rate found in our tests are not sufficient for determining the position of a pattern in a video for Augmented Reality applications in real time. One way to overcome this problem is using hybrid approaches, for example, optical flow techniques after the recognition of objects of interest, to maintain a frame rate best suited for this type of application.

For future work, we propose to analyze the use of tracking algorithms such as optical flow combined with the results obtained in this study. Applications that use similar approaches are not new in literature ([24]) but the analyses of those algorithms using a well known library as OpenCV can provide an interesting contribution.

References

1. R. Azuma, Y. Baillot, R. Behringer, S. Feiner, S. Julier and B. MacIntyre (2001), *Recent advances in augmented reality*, IEEE Computer Graphics and Applications, Vol.21, N.6, pp. 34-47.
2. I. E. Sutherland (1968), *A head-mounted three dimensional display*, Managing Requirements Knowledge, International Workshop on, Vol.0, pp. 757.
3. S. F. Kurniawan, I. K. G. Darma Putra and A. A. K. Sudana (2014), *Bone fracture detection using OpenCV*, Journal of Theoretical and Applied Information Technology, Vol.64, N.1, pp. 249–254.
4. A. Roy, N. Gale and L. Hong (2011), *Automated traffic surveillance using fusion of Doppler radar and video information*, Mathematical and Computer Modelling, Vol.54, N.1–2, pp. 531–543.

5. P. M. Panchal, S. R. Panchal and S. K. Shah (2013), *A Comparison of SIFT and SURF*, International Journal of Innovative Research in Computer and Communication Engineering, Vol.1, N.2, pp. 323-327
6. K. V. S. Bezerra and E. Aguiar (2013), *Casamento de padrões em imagens e vídeos usando características de imagens*, Workshop of Undergraduate Works (WUW) in SIBGRAPI 2013 (XXVI Conference on Graphics, Patterns and Images).
7. J. Heinly, E. Dunn and JM. Frahm (2012), *Comparative evaluation of binary features*, Computer Vision – ECCV 2012, Vol.7573, pp 759-773.
8. D. Mukherjee, Q. M. Jonathan Wu and G. Wang (2015), *A comparative experimental study of image feature detectors and descriptors*, Machine Vision and Applications, Vol.26, N.4, pp. 443–466.
9. E. Rublee, V. Rabaud, K. Konolige and G. Bradski (2011), *ORB: An efficient alternative to SIFT or SURF*, 2011 International Conference on Computer Vision (Barcelona), pp. 2564-2571.
10. R. Bouchiha and K. Besbes (2015), *Comparison of local descriptors for automatic remote sensing image registration*, Signal, Image and Video Processing, Vol.9, N.2, pp. 463–469.
11. E. Chatzilari, G. Liaros, S. Nikolopoulos and Y. Kompatsiaris (2013), *A comparative study on mobile visual recognition*, Machine Learning and Data Mining in Pattern Recognition: 9th International Conference, MLDM 2013, Proceedings, pp. 442-457.
12. O. D. Suarez, M. M. F. Carrobles, N. V. Enano, G. B. Garcia, I. S. Gracia, J. A. P. Incertis and J. S. Tercero (2014), *OpenCV essentials: acquire, process, and analyze visual content to build full-fledged imaging applications using OpenCV*, Packt Publishing Ltd (Birmingham).
13. E. Rosten and T. Drummond (2006), *Machine learning for high-speed corner detection*, Computer Vision – ECCV 2006, Lecture Notes in Computer Science, Vol.3951, pp. 430–443.
14. S. Leutenegger, M. Chli and R. Y. Siegwart (2011), *BRISK: Binary Robust invariant scalable keypoints*, International Conference on Computer Vision (Barcelona), pp. 2548-2555.
15. Jianbo Shi and C. Tomasi (1994), *Good features to track*, Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (Seattle), pp. 593-600.
16. C. Harris and M. Stephens (1988), *A combined corner and edge detector*, In Proc. of Fourth Alvey Vision Conference, pp. 147-151.
17. J. Matas, O. Chum, M. Urban and T. Pajdla (2004), *Robust wide-baseline stereo from maximally stable extremal regions*, Image and Vision Computing, Vol.22, I.10, pp. 761-767.
18. D. G. Lowe (1999), *Object recognition from local scale-invariant features*, Proceedings of the Seventh IEEE International Conference on Computer Vision (Kerkyra), Vol.2, pp. 1150-1157.
19. M. Agrawal, K. Konolige and M. R. Blas (2008), *CenSurE:Center Surround Extremas for realtime feature detection and matching*, Computer Vision – ECCV 2008, Lecture Notes in Computer Science, Vol 5305, pp 102-115.
20. H. Bay, T. Tuytelaars and L. Van Gool (2006), *SURF: Speeded Up Robust Features*, Computer Vision – ECCV 2006, Lecture Notes in Computer Science, Vol.3951, pp. 404-417.
21. M. Calonder, V. Lepetit, C. Strecha and P. Fua (2010), *BRIEF: Binary Robust Independent Elementary Features*, Computer Vision – ECCV 2010, Lecture Notes in Computer Science, Vol.6314, pp. 778-792.
22. A. Alahi, R. Ortiz and P. Vandergheynst (2012), *FREAK: Fast Retina Keypoint*, IEEE Conference on Computer Vision and Pattern Recognition (Providence), pp. 510-517.
23. M. Muja and D. Lowe (2009), *Fast approximate nearest neighbors with automatic algorithm configuration*, In VISAPP International Conference on Computer Vision Theory and Applications, Vol.1.
24. A. Ufkes and M. Fiala (2013), *A markerless augmented reality system for mobile devices*, International Conference on Computer and Robot Vision (Regina), pp. 226-233.