International Journal of Computer Applications Technology and Research
Volume 6–Issue 6, 245-248, 2017, ISSN:-2319–8656

# Object tracking with SURF: ARM-Based platform Implementation

H. Hassnaoui
EEA&TI laboratory, Hassan II University of Casablanca Faculty of Sciences and Techniques (FSTM)
Mohammedia, Morocco

A. Badri
EEA&TI laboratory, Hassan II University of Casablanca Faculty of Sciences and Techniques (FSTM)
Mohammedia, Morocco

A. Sahel
EEA&TI laboratory, Hassan II University of Casablanca Faculty of Sciences and Techniques (FSTM)
Mohammedia, Morocco

M. Akil
2ESIEE-Paris, 93162
Noisy-le-Grand Cedex, France

**Abstract**: Several algorithms for object tracking, are developed, but our method is slightly different, it's about how to adapt and implement such algorithms on mobile platform.
We started our work by studying and analyzing feature matching algorithms, to highlight the most appropriate implementation technique for our case.
In this paper, we propose a technique of implementation of the algorithm SURF (Speeded Up Robust Features), for purposes of recognition and object tracking in real time. This is achieved by the realization of an application on a mobile platform such a Raspberry pi, when we can select an image containing the object to be tracked, in the scene captured by the live camera pi. Our algorithm calculates the SURF descriptor for the two images to detect the similarity therebetween, and then matching between similar objects. In the second level, we extend our algorithm to achieve a tracking in real time, all that must respect raspberry pi performances. So, the first thing is setting up all libraries that the raspberry pi need, then adapt the algorithm with card's performances. This paper presents experimental results on a set of evaluation images as well as images obtained in real time.

**Keywords**: object tracking, mobile platform, feature matching, SURF, Raspberry pi

## 1. INTRODUCTION

In the field of computer vision, we seek the improvement of perception and visual recognition, by studying the algorithms carried out in this sense, and thus propose an adequate technique of implementation. Several techniques have been discussed to improve artificial vision. The different methods are based on image's content analysis, to extract the interest areas for studying. The descriptors of its zones are calculated independently on the scale (scale invariant) and rotation, to have the necessary information, which will be exploited to compare the images. That way we can detect and track objects. In the literature, the first method introducing the notion of independence at scale and rotation is that proposed by researcher David Lowe in 1999, called SIFT (Scale Invariant Features Transform). It is about detecting zones in an image which are known as interest points by DOG (Difference Of Gaussians) method and then, for each point, a descriptor vector of 128 dimensions is computed which is set The relation of this pixel with its neighborhood in the different scales or resolutions. This method is robust but has a major disadvantage that resides in the calculation time which is important which influences detection in real time. To remedy this problem, in 2006 researchers "ETH Zurich and Katholieke Universiteit Leuven" proposed an accelerated technique inspired by the SIFT which named SURF (Speeded Up Robust Features). SURF is based on Haar 2D wavelet responses and uses the integral images. As a basic characteristic, SURF uses a Haar wavelet approximation of blob detector based on the determinant of the Hessian matrix. These two methods are used for object detection or 3D reconstruction. Our work consists in proposing a method of implementing these techniques in a mobile platform while improving the calculation time to adapt the algorithm to such a platform. First, we will present the two methods (SIFT and SURF), after we will propose our technique of implementation in a mobile platform respecting the software architecture of the platform and its physical performances.

## 2. RELATED RESEARCHES

The first work on objects recognition is begun by the interest points extraction. An interest point is a point where the contour direction of an object changes abruptly (corner) or an intersection between two (or more) contour segments.

Moravec [1] considered a window as a neighborhood of the pixel, then determine the mean changes of the intensity in the neighborhood considered, moving the window in various directions. One of the main problems with this operator is that is not isotropic: If an edge is present that is not in the direction of the neighbors, then the edge will be badly chosen as an interest point. So, the Moravec operator is sensitive to noise. To remedy the problem of anisotropy Harris [2] proposes using a Gaussian filter W rather than a binary filter (0 or 1) used by Moravec. By these two methods it is possible to detect only the objects which have the same resolution and same angle of rotation. To solve this problem D. LOWE [5] proposes a robust algorithm called SIFT (Scale Invariant Features Transform) that is invariant with scale and rotation. This technique has the disadvantage of computing time which is important hence the invention of a new SURF [11] method inspired by the SIFT but it is three times faster.

# 3. FEATURE MATCHING

## 3.1 SIFT

The first step of the algorithm is interest points detection, called key points. A key point (x, y, σ) is defined by its coordinates on the image (x, y) and by its characteristic scale factor (σ). This is a circular interest zone; which radius is proportional to the scale factor. Each key point associated with an intrinsic orientation, dependent on the local content of the image around the key point, with the scale factor considered.

Detection and retrieval of features of interest points are carried out in four steps:

• Detection of extrema of scale-space

• Location of interest points,

• Choice of descriptor orientation,

• Calculation of descriptors

## 3.2 SURF

SURF detector is based on the Hessian matrix because of its good performance in computation time and accuracy [11]. Given a point x = (x, y) in an image I, the Hessian matrix H(x,σ) in x at scale σ is defined as follows:

$$H(x,\sigma) = \begin{pmatrix} L_{xx}(x,\sigma) & L_{xy}(x,\sigma) \\ L_{xy}(x,\sigma) & L_{yy}(x,\sigma) \end{pmatrix}$$

Where Lxx(x, σ) is the convolution of the Gaussian second order derivative ∂2/∂x2 g(σ) with the image I in point x, and similarly for Lxy(x, σ) and Lyy(x, σ). In practice, however, the Gaussian needs to be discretized and finite. On the next image you can see the partial derivatives of the Gaussian. First finite and discretized (the two left images) and then approximated by a 'box filter' in the directions y and xy. The gray areas are zero.

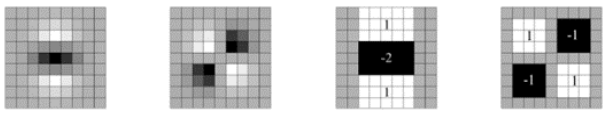


Figure 1: partial second derivatives of the Gaussian [11]

The approximation of the determinant of the Hessian matrix calculated in a point x of the image is stored in a “blob response map” and then local maxima are searched to find blobs.

It is interesting to find different scales to interest points to make the detector invariant to scale changes. This is often taken into account by creating a pyramid of images.

Each level of the pyramid represents a different scale. SURF can proceed differently through the box filters. Instead of successively applying the same filter to the output image filtered and sub-sampled, we can use box filters of various sizes directly on the original image. The “blob response maps” to different scales are constructed by enlarging the filter rather than repeatedly reducing the image size. This allows one hand to reduce the computation time and also prevent aliasing due to under - sampling of the image. The left image of the figure below shows the classical method with sub-sampling filter and constant size. On the right image filters vary in size.

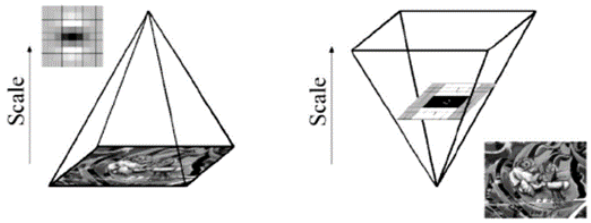


Figure 2: multiscale detection [11]

Scale spaces in SURF are implemented by applying box filters of different sizes. The output of the above 9×9 filter is considered as the initial scale layer, to which we will refer as scale s=1.2 (corresponding to Gaussian derivatives with σ=1.2). Specifically, this results in filters of size 9×9, 15×15, 21×21, 27×27, etc.

In searching for the maxima of the “blob response map” at different levels; we can now extract the position and size of the blobs in the image.

*Local neighborhood descriptor:*

The goal of a descriptor is to provide a unique and robust description of an image feature by describing the intensity distribution of the pixels within the neighbourhood of the interest point. A description is obtained for every interest point identified previously.

*Orientation:*

To achieve rotational invariance, the orientation of the interest point needs to be found. Bay et al. used The Haar wavelet responses in both x and y directions within a circular neighborhood of radius 6s around the interest point are computed, where s is the scale at which the interest point was detected. The dominant orientation is estimated by calculating the sum of all responses within a sliding orientation window of size π/3. The horizontal and vertical responses within the window are summed. The two summed responses then yield a local orientation vector. The longest such vector overall defines the orientation of the interest point.

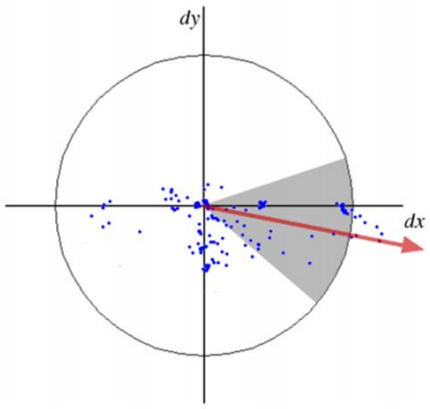


Figure. 3: Orientation assignment: A sliding orientation window of size π/3 detects the dominant orientation of the Gaussian weighted Haar wavelet responses at every sample point within a circular neighbourhood around the interest point. [11]

By comparing descriptors obtained from different images, *matching* pairs can be found.

# 4. IMPLEMENTATION

## 4.1 Raspberry pi

The Raspberry Pi is a single-board nano-computer with ARM processor, it allows the execution of several variants of the free operating system GNU / Linux and compatible software. In 2006, the first prototypes of the Raspberry Pi were developed on Atmega 644 ATmel microcontrollers. Different versions appeared: Model A, A +, B, B +, PI 2 and PI 3. The following table presents some technical characteristics of the pi version 3.

**Table 1. Technical characteristics of Raspberry Pi 3**

| technical characteristics of Raspberry Pi 3 | |
|---|---|
| **Input/output interface** | **Hardware Performance** |
| • 4 2.0 USB ports<br>• 10/100 Mb Ethernet<br>• HDMI<br>• Audio Outputs (3.5 mm phone jack)<br>• Storage: Micro SDHC slot<br>• Power supply: micro USB<br>• 40 I/O GPIO (General Purpose Input Output) with I2C, SPI, S2C, PWM… | • CPU: 1.2 GHz 64/32-bit quad-core ARM Cortex-A53<br>• 1 Go RAM<br>• GPU BCM videocore 4 full HD 1080p 30 fps |

## 4.2 Application

Our contribution is to adapt the matching algorithms to the Rasperry pi in real time. First, we looked for a suitable programming method for the card, installing all necessary libraries, we used the camera pi 8 Mp to acquire the images. Our objective is to find similar objects in the different images, so first, we must load or capture the image, containing the object to be detected or to follow, on which we apply the SURF algorithm to extract the interest points and then calculate the descriptors for these points. Then the same algorithm (SURF) will be applied on a scene image which we want to examine to know if the object in question exists or not. Finally, we apply the matching algorithm between the similar descriptors and thus between the similar areas in the two images (FIG. 4). To extend our application in real time we add a small improvement. Indeed, we define a video capture loop with a rate of 15 fps witch the SURF algorithm is applied to each of these frames (FIG. 5).

Our problem lies in the real-time operation which causes an overflow problem, because we overcome the performance of the card, to remedy it we deactivated the rotation parameter of SURF. Therefore, the algorithm is deprived now of the invariance to the rotation.

# 5. FIGURES

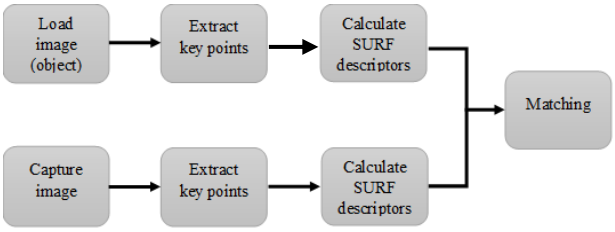


Figure. 4: object detection algorithm: captured image

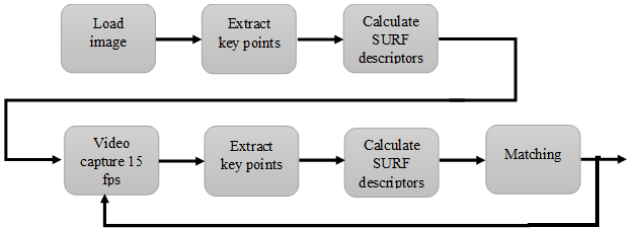


Figure. 5: object detection algorithm in real time

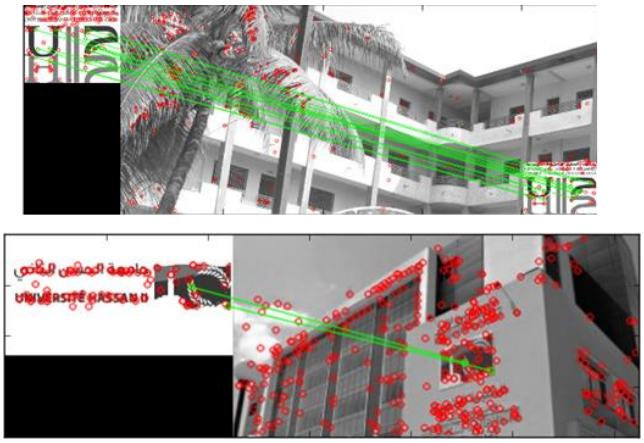

Figure. 6: object detection and matching using Pi camera and Raspberry pi 3

# 6. CONCLUSION

We have presented a technique of implementation of SURF algorithm in Raspberry pi, in order to detect objects in real time, with CPU computing on ARM-Based platform (Raspberry pi). Our algorithm brings together efficiency, speed and also portability. As such it is possible to optimize the artificial vision in the industrial sector.

This application has given a satisfactory results. In future work we will aim at optimizing the algorithm by parallel computing on GPU in order to combine SURF with HDR imaging in such platform.

## 7. REFERENCES

[1] Moravec, H. 1980. Obstacle avoidance and navigation in the real world by a seeing robot rover. In tech. report CMU-RI-TR-80-03, Robotics Institute, Carnegie Mellon University and doctoral dissertation, Stanford University, number CMU-RI-TR-80-03.

[2] Harris, C. and Stephens, M. 1988. A combined corner and edge detector. In Proc. Fourth Alvey Vision Conference, pages 147–151.

[3] Lindeberg, T. 1988. Feature detection with automatic scale selection. IJCV 30(2).

[4] Mikolajczyk, K. 2002. Interest point detection invariant to affine transformations. PhD thesis,Institut National Polytechnique de Grenoble.

[5] Lowe, D.G. 1999. Object recognition from local scale-invariant features. In Proceedings of the 7th International Conference on Computer Vision, Kerkyra, Greece, pp. 1150–1157.

[6] David, G. Lowe, 2004. Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision, vol. 60, no 2, 2004, p. 91–110.

[7] Brown, M. and Lowe, D.G. 2002. Invariant features from interest point groups. In British Machine Vision Conference, BMVC 2002, Cardiff, Wales, pp. 656-665.

[8] Lindeberg, T. and Bretzner, L. 2003. Real-time scale selection in hybrid multi-scale representations. In Proceedings of the 4th international conference on Scale space methods in computer vision, vol. 2695, Berlin, Springer-Verlag.

[9] Lowe, D.G. 2001. Local feature view clustering for 3D object recognition. IEEE Conference on ComputerVision and Pattern Recognition, Kauai, Hawaii, pp. 682-688.

[10] Ke, Y. and Sukthankar, R. 2004. PCA-SIFT: A More Distinctive Representation for Local Image Descriptors. Proc. Conf. Computer Vision and Pattern Recognition, pp. 511-517.

[11] Bay, H., Ess, A., Tuytelaars, T. and Van Gool, L. 2008. SURF: Speeded Up Robust Features. Computer Vision and Image Understanding (CVIU), Vol. 110, No. 3, pp. 346—359.

[12] Fan, X. et al. 2013. Implementation of high performance Hardware architecture of OpenSURF Algorithm on FPGA. International Conference on Field-Programmable Technology (FPT). IEEE Conference Publications.

[13] Wilson, C et al. 2014. A Power-Efficient Real-Time Architecture for SURF Feature Extraction. International Conference on ReConFigurable Computing and FPGAs (ReConFig14). IEEE Conference Publications.

[14] Chao, J., Huitl, R., Steinbach, E. and Schroeder, D. 2015. A novel rate control framework for sift/surf feature preservation in h.264/avc video compression. IEEE Transactions on Circuits and Systems for Video Technology, vol. 25, no. 6, pp. 958–972.

[15] Chao, J et al. 2015. A Novel Rate Control Framework for SIFT/SURF Feature Preservation in H.264/AVC Video Compression. IEEE Transactions On Circuits And Systems For Video Technology, Vol. 25, no. 6.

[16] Chen, W. et al. 2016. FPGA-Based Parallel Implementation of SURF Algorithm. IEEE 22nd International Conference on Parallel and Distributed Systems (ICPADS) Pages: 308 – 315.