A comparison of feature extractors for panorama stitching in an autonomous car architecture.

CORTÉS-GALLARDO, E., MORENO-GARCIA, C.F., ZHU, A., CHÍPULI-SILVA, D., GONZÁLEZ-GONZÁLEZ, J.A., MORALES-ORTIZ, D., FERNÁNDEZ, S., URRIZA, B., VALVERDE-LÓPEZ, J., MARÍN, A., PÉREZ, H., IZQUIERDO-REYES, J. and BUSTAMANTE-BELLO, R.

2019

A Comparison of Feature Extractors for Panorama Stitching in an Autonomous Car Architecture

Edgar Cortés-Gallardo[1], Carlos Francisco Moreno-Garcia[2], Alfredo Zhu[1], Daniela Chípuli-Silva[1], José A. Gonzalez-González[1], Domenico Morales-Ortiz[1], Sebastián Fernández[1], Bernardo Urriza[1], Juan Valverde-López[1], Arath Marín[1], Hugo Pérez[1], Javier Izquierdo-Reyes[1,3], Rogelio Bustamante-Bello[1,4]
A01336292@itesm.mx, c.moreno-garcia@rgu.ac.uk, {A01651980, A01652237, A01652551, A00820324, A01652293, A01336299, A01656127, A01651107, A01337226}@itesm.mx, {jizquierdo.reyes, rbustama}@tec.mx
[1]Tecnológico de Monterrey, Escuela de Ingeniería y Ciencias, Campus Ciudad de México, México
[2]The Robert Gordon University, Aberdeen, United Kingdom
[3]Massachusetts Institute of Technology, Massachusetts, United States
[4]Centro de Investigación en Microsistemas y Biodiseño, Tecnológico de Monterrey, México

Abstract— Panorama stitching consists on frames being merged to create a 360° view. This technique is proposed for its implementation in autonomous vehicles instead of the use of an external 360-degree camera, mostly due to its reduced cost and improved aerodynamics. This strategy requires a fast and robust set of features to be extracted from the images obtained by the cameras located around the inside of the car, in order to effectively compute the panoramic view in real time and avoid hazards on the road. This paper compares and creates discussion of three feature extraction methods (i.e. SIFT, BRISK and SURF) for image feature extraction, in order to decide which one is more suitable for a panorama stitching application in an autonomous car architecture. Experimental validation shows that SURF exhibits an improved performance under a variety of image transformations, and thus appears to be the most suitable of these three methods, given its accuracy when comparing features between both images, while maintaining a low time consumption. Furthermore, a comparison of the results obtained with respect to similar work allows us to increase the reliability of our methodology and the reach of our conclusions.

Keywords — Panorama Stitching, Image Blending, Feature Extraction, Autonomous Vehicles, SIFT, BRISK, SURF.

I. INTRODUCTION

Blind spots are a very common downside in everyday drivers' routines. The most common example is the rear-view mirrors, which are used to keep watch on cars coming from behind. These offer a certain visual perspective, nonetheless they do not reflect all the objects that are behind the car. The human eye gives a peripheral field of around 135- to 200-degrees, but a regular camera has a field of view of only 35- to 50-degrees. Due to this lack of sight, some accidents may happen on the road. Regarding this safety issue, autonomous cars that are being developed for driving assistance see the outside of the car using strategically located cameras. Therefore, panoramic image stitching works by taking several pictures from an ordinary camera and blending them together to produce a single image with a much larger line of vision [1].

However, the blind spot problem persists, as a single camera cannot perceive the 360-degree perspective of the car by itself. Although there are existing 360-degree cameras on the market, these are too expensive, and given that they must be situated outside the vehicle's body, they become prone to theft [2]. Moreover, these cameras are aerodynamically inefficient since air currents causing drag in the vehicle tend to affect in larger scales the development of vehicles speed [3]. This is given by

$$F_d = \frac{1}{2} C_d \rho A V^2 \quad (1)$$

which states that the drag force F_d is equal to half of the drag coefficient C_d, which depends on the object's geometry; in this case, the vehicle itself. This value, multiplied by the fluid density ρ, the colliding area A (i.e. the area of the vehicle perpendicular to air trajectory) and the object's velocity relative to air direction squared V^2 imply that the drag force experienced by an object will be proportional to the speed it has relative to air direction. Therefore, adding an external camera makes the vehicle cover a wider area, which will affect the drag force as stated previously.

To address the paradigm of coordinating the vision from more than one visual input, some authors have proposed different approaches that create a joint view based on multiple cameras. Cortés et al. [4] proposed a semi-automatic pose estimation method for a fleet of robots with stereoscopic cameras by means of an interface which allows a human expert to correct and impose mappings between two images. This method allows the robots to align and follow a common route without requiring a GPS or landmark application. Moreover, Manzo et al. [5] presented an interactive pose calibration method for a set of cameras used in video surveillance. The scheme of the human assisted interface consists of a set of cameras with salient points and feature extractors, followed by a matching estimator assisted by the user, a structure for motion and finally a cooperative pose estimation model capable of producing homographies.

Although these techniques are suitable for alignment and surveillance purposes, an implementation for a real time autonomous vehicle case would require an approach not relying on human assistance. One of the simplest, yet most viable solutions for the problem at hand is panorama stitching, where frames are taken from each of the cameras located around the inside of the car, creating a panoramic video stream of a 360-degree sight. This method has proven to be effective in some scenarios such as the automatization of panoramic image stitching and detection of multiple panoramas in a single image stream array [6]. Another application involving automotive development would be panoramic image stitching of rear cameras instead of rear mirrors in cars also known as rear-stitched view panorama [7], as well as the aforementioned 360-degree panorama stitching approach to avoid both blind spots and a 360-degree camera [8]. To perform panorama stitching, it is fundamental to select an image feature extraction method which is robust in terms of accuracy and time of computation. From the literature [9], we have observed that *SIFT*, *BRISK* and *SURF* are the most commonly used for similar tasks, and thus will be discussed in this paper.

This paper is organized as follows. *Section 2* presents the potential methods for feature extraction in order to choose the method that is best suited for real time panorama image stitching. *Section 3* analyzes the methodology used for the method discrimination. It is clear to state that the methodology involves both: theoretical explanation of the method's implementations and practical code application for actual real time comparison between methods and algorithms. Moreover, *Section* 4 presents the results obtained and shows a comparison with the state of the art. Finally, *Section 5* is reserved for the conclusions as well as the intended future developments.

II. FEATURE EXTRACTION

This section introduces the three methodologies selected to perform image feature extraction for panorama stitching in a set of cameras located outside the autonomous vehicle body: *SIFT*, *SURF* and *BRISK*.

2.a SIFT

SIFT (Scale Invariant Feature Transform) is a method that determines salient points. Proposed by Lowe et al. [10], it has four computational steps for extracting keypoints: scale-space peak selection, keypoint localization, orientation assignment and defining keypoint descriptors. For each image, it builds an image pyramid by generating progressively blurred out images, and it subtracts neighbor images to get the Difference of Gaussian (DOG) pyramid. Then, it detects the extreme for DOG pyramid. The number of keypoints was reduced to help in increasing efficiency and robustness of the technique. Keypoints are rejected if they had a low contrast or if they were located on an edge. The following step is orientation assignment, which uses an orientation histogram to statistic the gradient orientation by sampling the center neighborhood of the key points. The last step consists on obtaining the keypoint descriptors [11]. In other words, it uses the scale and local orientation by maximizing the difference of Gaussian in scale and space. With this information, it computes a gradient orientation histogram for each cell with eight orientations to obtain the dimensional descriptor. Then, it normalizes the descriptor to obtain an invariant to intensity change. This process is done for all keypoints.

2 b. BRISK

BRISK (Binary Robust Invariant Scalable Keypoints) is a method that solves the problem of classical computer vision detection, which matches image key points without sufficient prior knowledge in the field and camera position. It was proposed by Leutenegger et al. [12], it detects the corners and then it filters them with the FAST (Feature from Accelerated Segment Test) approach [13]. BRISK identifies the characteristic direction of the features to achieve rotation invariance. The descriptor is constructed as a binary string and the features are invariant to scale, limited affine changes and rotation.

BRISK is an algorithm for feature point detection and description with scale invariance and rotation invariance. The principle of it is to extract the stable extreme points of sub-pixel precision in the scale space pyramid constructed. It can find random point pairs neighboring the local image by using the gray scale relationship and obtaining the binary feature descriptor of each keypoint. The difference between it with others is that BRISK does not require high storage memory and it is faster, but it implies reducing the robustness [14].

2 c. SURF

SURF (Speeded Up Robust Features) is an algorithm developed for local, similarity invariant representation and comparison [15]. It approximates Gaussian smoothing with box filters; this technique allows the image filtering to be faster if the whole image is used. SURF is composed of three main steps. First, keypoints are selected at distinctive locations in the image, such as corners, blobs, and T-junctions. Next, the neighborhood of every keypoint is represented by a feature vector. This descriptor has to be distinctive. At the same time, it should be robust to noise, detection errors, and geometric and photometric deformations. Finally, the descriptor vectors are matched among the different images [11].

SURF uses the Hessian matrix as a blob detector to find the keypoints; thus, the determinant of this matrix is used to measure the local change with the neighborhood points and the ones with the maximal value are chosen and are interpolated in scale and image space. In order to obtain the rotational invariance, it finds the orientation of the point using the sum of Haar wavelet responses. Finally, it compares the descriptor obtaining the match [15]. It's important to mention that square shaped filter is used as an approximation.

If the integral image is processed, then the square shaped filters provide the best result [16]. The integral image describes the sum of pixels to the left and above a specified pixel and represents the average values of pixels over a certain region. This is important because the values of the integral image are used to compute the Hessian matrix, as they are much faster for doing the convolution needed to obtain$L_{xx}(X,\sigma)$. An integral image can be computer for any point x, y as follows:

$$S(x,y) = \sum_{i=0}^{x} \sum_{j=0}^{y} I(i,j) \qquad (2)$$

As mentioned before, the SURF detector is based on the determinant of the Hessian matrix. Let point $X = (x,y)$ in an image I, the Hessian matrix $H(X,\sigma)$ at scale σ in X can be calculated as in matrix:

$$H(X,\sigma) = \begin{bmatrix} L_{xx}(X,\sigma) & L_{xy}(X,\sigma) \\ L_{xy}(X,\sigma) & L_{yy}(X,\sigma) \end{bmatrix} \qquad (3)$$

where, $L_{xx}(X,\sigma)$ is the convolution of the Gaussian second order derivative $\frac{\partial^2}{\partial x^2} g(\sigma)$ with the image I in the point X, and similarly for $L_{xy}(X,\sigma)$ and $L_{yy}(X,\sigma)$.

The implementation specifications are the following: Code was executed in an ASUS Zephyrus computer, with i7 8th Gen Intel core, 40GB RAM memory and the NVIDIA Geforce RTX 2080 GPU running C++ compiled in Ubuntu 18.04 operative system. OpenCV library was used for image processing tasks.

III. METHODOLOGY

3.a. Selection of Feature Extractors

The application of the three methods is code based, which means that in order to obtain, compare and analyze results, the implementation of the algorithms in actual code would be required. On this regard, SIFT was analyzed only from a literature review perspective due to the following reason. Given that our main goal is to execute these feature extraction methods in a video stream, the panorama stitching algorithm needs to be fast, and with a precise image processing transform. SIFT is precise, but lacks velocity due to its high computational demand, therefore at this point it is discarded from further analysis in this paper. By contrast, there is no mention in literature regarding a high computational cost of BRISK, scale, rotation and affine invariant, using scale pyramids maxima and corners for feature extraction, which is a very good approximation. Furthermore, SURF is regarded in literature to be superior in terms of execution time and illumination invariance, having a low computational cost in balance with fair precision, by means of Gaussian pyramids and Hessian matrices applied for acquiring speed. For comparison with more recently presented feature extractors in literature, it is worth mentioning that methods such as ORB, which is the fast and rotation invariant version of the Binary Robust Independent Elementary Features (BRIEF) algorithm [17], is rotation and scale invariant with improved execution time, but its performance is poorer in the presence of noise compared to the selected feature extractors [18].

3.b. Panorama Stitching based on SURF

In comparison to SIFT, as mentioned in literature, SURF is used as the feature descriptor and for matching purposes. A basic second order Hessian matrix approximation is used for feature point detection. The time needed to generate the output is 40 seconds [19].

1. In the construction of scale image pyramid in SURF algorithm, the scale space is divided into octaves, and there are 4 scale levels in each octave.
2. Each octave represents a series of filter response maps obtained by convolving the same input image with a filter of increasing size.
3. The minimum scale difference between subsequent scales depends on the length of the positive or negative lobes of the partial second order derivative in the direction of derivation.

4. A non-maximum suppression is done using a neighborhood of $3 \times 3 \times 3$ to get the steady feature points and the scale of values.

3.c. Panorama Stitching based on BRISK

BRISK detects corners using the Adaptive Generic Accelerated Segment Test (AGAST) algorithm and filters the results through the FAST Corner Score. The following steps are carried out:

1. Corners are used to detect and search for maxima in every reduction of the scale space pyramid method.
2. In construction, BRISK descriptors identify the characteristic direction of each feature vector.
3. This way, feature invariance can be achieved and therefore make processes for rotations.
4. A binary string is constructed for brightness tests and achieve illumination invariance.

3.d. Implementation and Results

As shown in *Figure 1* and *Figure 2*, for BRISK and SURF implementations, both appear to have a precise point allocation for image feature extraction. BRISK allocates a higher number of points in comparison to SURF, however this is not enough evidence of allocation precision. When stitching up both images, it can be seen in *Figure 3* and *Figure 4* that both algorithms have an adequate image reconstruction when blending them together. Literature [20] supports the fact that both methods seem to have similar accuracy for keypoint detection and descriptors, thus, feature extraction efficiency is not the main issue for this problem's solution. The main issue for discussion then becomes the speed of the algorithm.

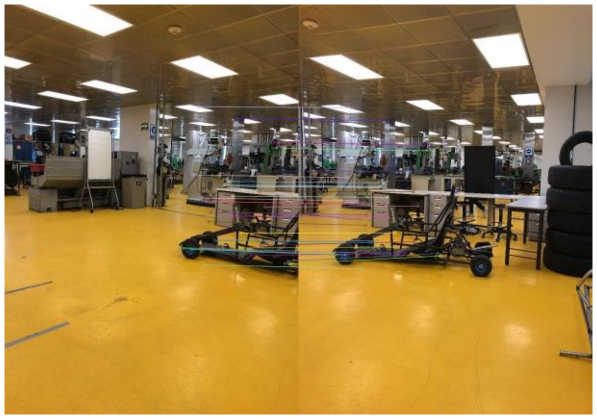


Figure 1. SURF Implementation.

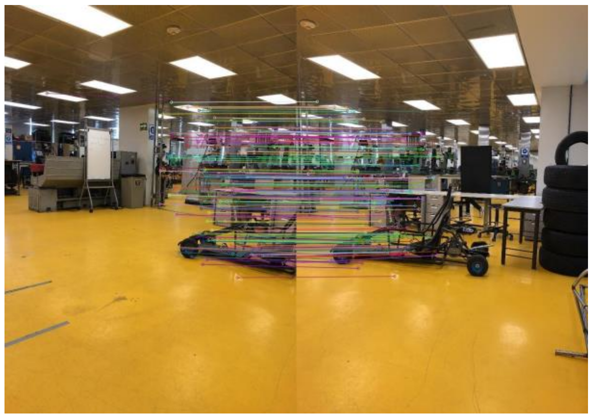


Figure 2. BRISK Implementation

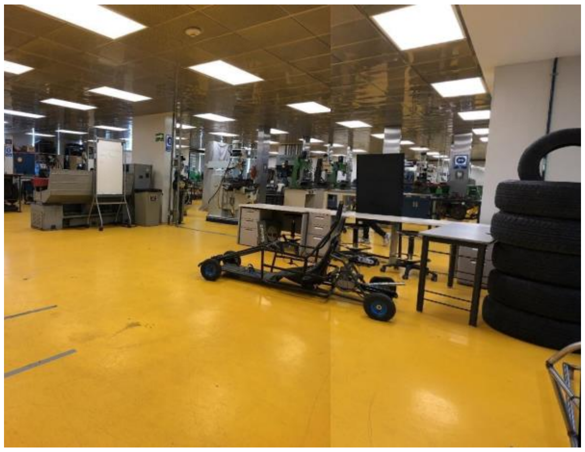


Figure 3. SURF Stitching

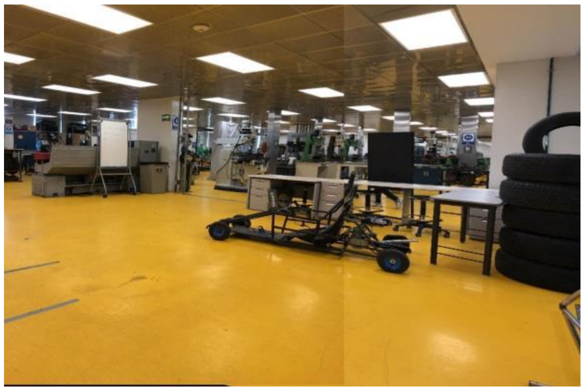


Figure 4. BRISK Stitching.

When comparing time of execution of both algorithms at the time of allocating the points from the feature extraction methods and blending both images together for panorama stitching and reconstruction, SURF’s time is two times faster than BRISK’s. More specifically, BRISK takes an average of 0.4849 seconds to reconstruct the final product of the image processing stitching. As for SURF, it takes an average of 0.2174, making it far more efficient for the needs of our system.

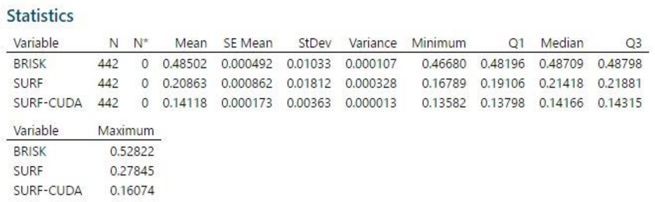
Some authors have confirmed similar results when comparing runtime needed for recognizing matches using the mentioned methods. For instance, Juan et. al [1] reported that SURF presented a much faster runtime in the process of detection and matching when compared to SIFT and PCA SIFT (which consists of normalizing the gradient patch instead of using an orientation histogram). Meanwhile, SIFT proved to be more efficient in matching keypoints, due to the differences in lighting within the image they used. In other work by the same authors [21], SURF proved to be several orders of magnitude faster. Also, Karami et al. [22] performed similar comparisons using SIFT, SURF, BRIEF and ORB. Since BRIEF consists of a less complex SIFT variation and ORB creates a rotation matrix for the image using a BRIEF descriptor, SIFT once again proved to be the slowest method, while SURF proved to take the same average time as ORB for identifying images with varying light intensity. However, ORB had the lowest matching rate among them all. In special scenarios where there is no rotation invariance and different orientations, ORB performed better, but overall SURF provided the best balance between speed and accuracy.

For the resolution of this problem, it appears that SURF is the fastest approach, but not fast enough due to the fact that it is computed inside a CPU. Therefore, we propose to accelerate the process by means of parallelization by running the code using a GPU architecture. Parallelization is implemented using CUDA 10.0 toolkit for CUDA. C/C++ application is needed, using OpenCV parallelized libraries for accelerating the SURF computation and gain even more speed without having a downgrade in feature extraction efficiency [23]. Experiments showed that this implementation did accelerate the process; in fact, it halved the execution time, taking only 0.1412 seconds to deliver the same output shown in Figure 3.

IV. Results

Figure 5 shows that SURF blends both images in half the time compared to BRISK. This is the main reason for parallelizing SURF with CUDA instead of BRISK. Moreover, Figure 6 shows that BRISK and SURF have an abnormal step in iteration 129. This happens because BRISK and SURF are running inside the CPU architecture. This is also due to the thermal throttling technique, which is in charge of regulating the thermal environment of the microprocessor by reducing the speed of the device and entering a ventilation state. Since the CPU is in charge of all processes inside the machine, its integrity is essential [24]. In contrast, CUDA SURF runs inside the GPU architecture, which is not overseen by the CPU. Because of this, there's no step when developing the CUDA SURF through an extended number of iterations. Moreover, in Figure 7 it is validated that CUDA SURF has the least standard deviation compared to the other methods, implying that it will have more constant display than BRISK and CPU performed SURF. BRISK counts with a 0.01033 standard deviation, as for SURF being very similar with a 0.01812 ratio. SURF CUDA on the other hand, as a 0.00363 standard deviation value, putting it on top of the other methods. The minimum processing time period for BRISK was of 0.46680 seconds, SURF had a 0.16789 seconds and SURF CUDA was of 0.13582 seconds. And the maxima for stitching time values for BRISK, SURF and SURF CUDA was: 0.52822, 0.27845 and 0.16074 seconds respectively. This analysis was made from a 442 data samples in all three methods. As it can be seen, SURF CUDA was the fastest method implemented of all three. Since processing requires CPU and GPU cores, time taken for stitching up images is not always the same, and SURF CUDA is not only the fastest method, but also the one that has a much less variance ratio between iterations, by far. All these statements can be referenced in Table 1. SURF CUDA can be used not only for panorama stitching, but its development in literature and practice suggests that it can be used as one of the key feature extractors with a very promising roll for real time applications.

TABLE I
Statistical Data of BRISK, SURF and CUDA SURF


Statistics

Variable	N	N*	Mean	SE Mean	StDev	Variance	Minimum	Q1	Median	Q3
BRISK	442	0	0.48502	0.000492	0.01033	0.000107	0.46680	0.48196	0.48709	0.48798
SURF	442	0	0.20863	0.000862	0.01812	0.000328	0.16789	0.19106	0.21418	0.21881
SURF-CUDA	442	0	0.14118	0.000173	0.00363	0.000013	0.13582	0.13798	0.14166	0.14315

Variable	Maximum
BRISK	0.52822
SURF	0.27845
SURF-CUDA	0.16074

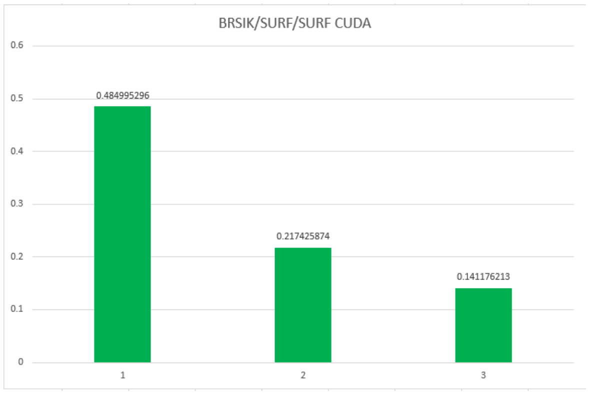


Figure 5. BRISK, SURF and CUDA SURF runtime comparison in seconds.

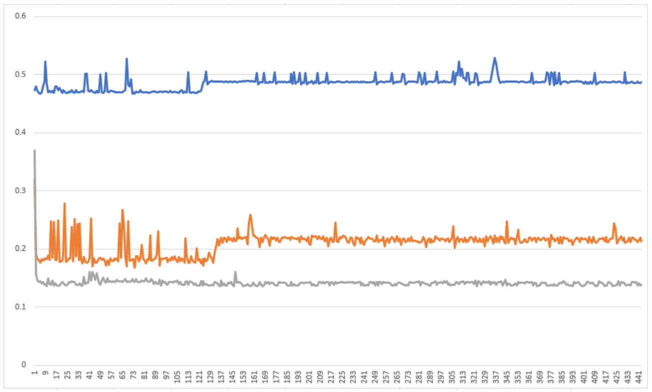


Figure 6. BRISK, SURF and CUDA SURF development runtime in seconds.

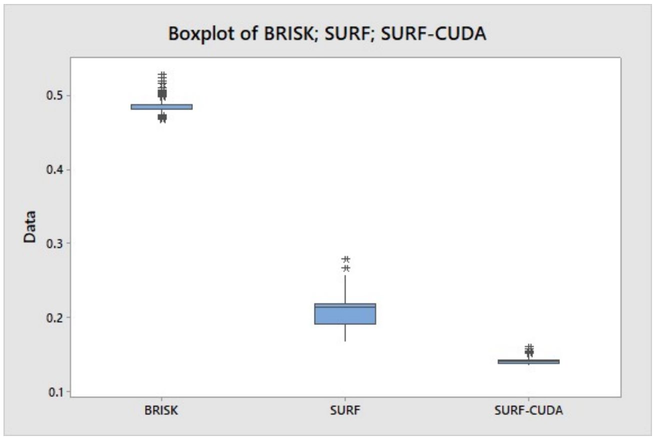


Figure 7. BRISK, SURF and CUDA SURF percentile box plot.

V. Conclusions

Image mosaicking/stitching is an active research area in the fields of computer vision and computer graphics. To these aims, there is a handful of different algorithms for feature detection and extraction. The choice of the feature detector/extractor depends on the problem at hand, however we can get some intuitions of which algorithm could have a higher affinity with any given scenario. In this work, we have tested a number of feature extraction algorithms for its use in panorama stitching for autonomous vehicles.

As shown in the experimental validation, the runtime comparison between three different methods allows us to conclude that SURF is the most time efficient between the feature extraction methods proposed by literature. Moreover, it is shown that parallelization using CUDA proved to be an essential requirement for keeping processing time to its minimum. This is not only due to the speed-up that comes inherently from parallelizing, but also given the thermal throttling that occurs to the CPU when not parallelizing, which slows down the process.

However, it is worth noting that these methods are not the only ones available for these purposes. Image stitching techniques are constantly evolving, and new alternatives are continuously being created. As computer's processing power continues to grow, so does the importance and application possibilities in computer vision.

It is important to have accurate timing measurements, particularly for this kind of application in which the reaction time of an autonomous automobile sometimes needs to take a fraction of a second to prevent accidents in which a human being could be hurt. Having a small processing timing isn't just an efficiency parameter, but a safety requirement as well.

The comparison of these different imaging stitching techniques allows to generate more documentation about their efficiency and provides project developers the means to improve their designs and promote the advancement of computer vision technology. Moreover, we must keep in mind that panorama stitching is not the only goal of this feature extraction methods; so further research and implementations need to be explored in pursuit of more innovative solutions for everyday requirements.

Acknowledgment

Authors thank to Centro de Investigación en Microsistemas y Biodiseño (cimb.com.mx) at Tecnológico de Monterrey Campus Ciudad de México for all the support in the research process.

References

[1] Juan, L. Gwun, O., “SURF applied in panorama image stitching,” International Conference on Image Processing Theory, Tools and Applications (IPTA), pp. 495-499, 2010.

[2] Cheng, K.W.E., “Recent development on electric vehicles,” International Conference on Power Electronics Systems and Applications (PESA) pp. 1-5, 2009.

[3] De Lima Bernardo, B., Moraes, F., Rosas, A. “Drag Force Experienced by a Body Moving through a Rarefied Gas,” Chinese Journal of Physics, vol. 51, no. 2, pp. 189-199 2013.

[4] Cortés, X., Serratosa, F., Moreno-García, C. F., “Semi-automatic pose estimation of a fleet of robots with embedded stereoscopic cameras,” in Emerging Technologies and Factory Automation (ETFA), pp. 1-6, 2016.

[5] Manzo, G., Serratosa, F., Vento, M., “Interactive pose calibration of a set of cameras for video surveillance,” Emerging Technologies and Factory Automation (ETFA), pp. 1-4, 2016.

[6] Brown, M., Lowe, D. G., “Automatic panoramic image stitching using invariant features,” International Journal of Computer Vision, vol. 74, no. 1, pp. 59-73.

[7] Janice, P., Vikram, A., Villareal, J., Weaver, L., Do-Kyoung, K., “Rear-Stitched View Panorama: A low-power embedded implementation for smart rear-view mirrors on vehicles,” Computer Vision and Pattern Recognition Workshops (CVPRW), 2017.

[8] Shah, A. A., Mustafa, G., Ali, Z., Anees, T., “Video stitching with localized 360 model for intelligent car parking monitoring and assistance system,” International Journal of Computer Science and Network Security (IJCSNS), vol. 19, no. 5, pp. 43-47, 2019.

[9] Tareen, S. A. K., Saleem, Z., “A comparative analysis of SIFT, SURF, KAZE, AKAZE, ORB, and BRISK,”. International Conference on Computing, Mathematics and Engineering Technologies (iCoMET), pp. 1-10, 2018.

[10] Lowe, D. G., “Distinctive image features from scale invariant keypoints,” International Journal of Computer Vision, vol. 60, pp. 91–110, 2004.

[11] Adel, E. & Elmogy, M. & El-Bakry, H., “Image Stitching based on Feature Extraction Techniques: A Survey,” International Journal of Computer Applications, vol. 99, pp. 1-8, 2014.

[12] Leutenegger, S., Chli, M., Siegwart, R., “BRISK: binary robust invariant scalable keypoints,” International Conference on Computer Vision (ICCV), pp. 2548-2555, 2011.

[13] Rosten, E., Porter, R., Drummond, T., “Faster and better: A machine learning approach to corner detection,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 32, no. 1, pp. 105–119, 2010

[14] Liu, Y., Zhang, H., Guo, H., Xiong, N. N., “A FAST-BRISK feature detector with depth information,” Sensors, vol. 18, no. 11, 3908, 2018.

[15] Bay, H., Ess, A., Tuytelaars, T., Van Gool, L., “SURF: speeded up robust features,” Computer Vision and Image Understanding, vol. 110, no. 3, pp. 346-359, 2008.

[16] Uk, I., “A detailed analysis on feature extraction techniques of panoramic image stitching algorithm,” International Journal of Engineering and Applied Computer Science, vol. 2, pp. 147-153, 2017.

[17] Calonder, M., Lepetit, V., Strecha, C., Fua, P., “BRIEF: binary robust independent elementary features,” European Conference on Computer Vision (ECCV), pp. 778-792, 2010.

[18] Khan, N.Y., McCane, B., Wyvill, G., “SIFT and SURF performance evaluation against various image deformations on benchmark dataset,” IEEE International Conference on Digital Image Computing Techniques and Applications (DICTA), pp. 501-506, 2011.

[19] Arya Mary, K. J., Priya, S., “Panoramic image stitching based on feature extraction and correlation,” National Conference on Future Technologies in Power, Control and Communication Systems (NFTPCOS-17), pp. 32-39, 2017.

[20] Schaeffer, C., “A Comparison of Keypoint Descriptors in the Context of Pedestrian Detection: FREAK vs. SURF vs. BRISK,” Stanford University, CS Department.

[21] Juan, L. Gwun, O., “A comparison of SIFT, PCA-SIFT and SURF,” International Journal of Image Processing, vol. 3, no. 4, pp. 143-152, 2009.

[22] Karami, E., Prasad, S., Shehata, M., “Image matching using SIFT, SURF, BRIEF and ORB: performance comparison for distorted images,” Newfoundland Electrical and Computer Engineering Conference, 2017.

[23] Daga, B., Bhute, A., Ghatol, A., “Implementation of Parallel Image Processing Using NVIDIA GPU Framework,” Advances in Computing, Communication and Control. (ICAC3), Communications in Computer and Information Science, vol. 125, 2011.

[24] Sulaiman, D. R., “Microprocessors thermal challenges for portable and embedded systems using thermal throttling technique,” Procedia Computer Science, vol. 3, pp. 1023-1032, 2011.