Image processing for feature detection and extraction

Nicolae APOSTOLESCU*,[1], Dragos-Daniel ION-GUTA[2]

*Corresponding author
*,[1]Aerospace Consulting,
B-dul Iuliu Maniu 220, Bucharest 061126, Romania,
apostolescu.nicolae@incas.ro
[2]INCAS – National Institute for Aerospace Research "Elie Carafoli",
B-dul Iuliu Maniu 220, Bucharest 061126, Romania,

DOI: 10.13111/2066-8201.2024.16.3.1

Received: 25 June 2024/ Accepted: 19 August 2024/ Published: September 2024

Abstract: *The present paper aims to conduct an experiment that compares different methods of detecting objects in images. Programs were developed to evaluate the efficiency of SURF, BRISK, MSER, and ORB object detection methods. Four static gray images with sufficiently different histograms were used. The experiment also highlighted the need for image preprocessing to improve feature extraction and detection. Thus, a programmed method for adjusting pixel groups was developed. This method proved useful when one of the listed algorithms failed to detect the object in the original image, but succeeded after adjustment. The effectiveness of detection methods and the evaluation of their performance depend on the application, image preparation, algorithms used, and their implementation. Results of the detection methods were presented numerically (similarities, gradients, distances, etc.) and graphically.*

Key Words: *feature descriptors, feature detector, image matching, SIFT, SURF, BRIEF, FAST, BRISK, ORB, MSER*

1. INTRODUCTION

The objective of this paper is to conduct an experiment on the behavior of digital methods for matching two images. One of the images represents an object contained in the second image, here called the scene. In the literature, the operation is called *matching* and ends with the detection of the object in the image [1-2]. In machining processes, the digital image is a matrix in which individual pixel processing does not provide information for interpreting the image but only for improving its visual appearance [3]. The pixel matrix can be processed to obtain relevant features represented by numerical values or descriptors that encode information found in different regions of the image [4]. Features are locations in the image with unique, repeatable structures and invariant to geometric transformations such as scaling, rotation or lighting changes. Features in an image are represented by numerical values or descriptors that encode information found in different regions of the image [5]. The feature extraction process involves analyzing pixel values and identifying specific patterns that can be used to represent the content of the image in a more compact and interpretive way [6]. Selecting and extracting features from an image are the most important steps in all computer vision applications (detecting, recognizing, and tracking objects in static or mobile images) [2]. Object detection has numerous applications in computer vision, such as object tracking, retrieval, video surveillance, image captioning, image segmentation, medical imaging, and several other

applications [7]. The image processing techniques such as BRISK, SURF, and ORB are crucial in enabling precise navigation, obstacle detection, and collision avoidance through visual data [1-2], [8-9]. The process of extracting features involves analyzing pixel values and identifying appropriate patterns, methods, and algorithms to represent image content in a more compact and interpretive way.

2. IMAGE PROCESSING AND FEATURE DETECTION TECHNIQUES

In the present experiment, histogram equalization techniques, adjusting image intensity values to a specified interval, and filtering were used to improve the image. Equalizing the histogram of an image involves obtaining a new image in which the pixel intensity values are within a certain range of values. Adjusting the image intensity values to a specified interval involves obtaining a new image according to the rule proposed in Table 1.

Table 1 – Condition on pixel adjusting value

Pixel value $v(x,y)$	New pixel value after adjustment
$v(x,y) \leq inf_{inp}$	inf_{out}
$v(x,y) >= sup_{inp}$	sup_{out}
$inf_{inp} < v(x,y) < sup_{inp}$	$inf_{out} + \frac{v(x,y) - inf_{inp}}{sup_{inp} - inf_{inp}} * (sup_{out} - inf_{out})$

Table 2 – Example of image adjustment

inf_{inp} = 51; sup_{inp} = 76;					inf_{out} = 102; sup_{out} = 127;				
234	197	45	155	69	127	127	102	127	120
1	83	185	49	196	102	127	127	102	127
118	201	121	189	49	127	127	127	127	102
109	121	39	62	74	127	127	102	113	125
118	10	87	234	24	127	102	127	127	102

The figures below, Fig. 1- 4, show an image to which histogram equalization and pixel intensity adjustments have been applied.

Selecting and extracting features from an image are the most important steps in all computer vision applications (detecting, recognizing, and tracking objects in still or moving images). The operations involve numerical transformations on the spatial representations of the images with the preservation of the essential information from the original images.

In the context of this paper, the components of feature detection and matching include describing, detecting, extracting, and matching features in images. The description looks at how certain information is associated around points of interest, such as gradient or intensity. Feature detection involves identifying structures, points, and regions of interest. The features provide unique information about the image. Feature matching consists of finding pairs of similar features from two or more images. To improve the accuracy of correspondences, it is common to apply special techniques, including geometric verification.

Feature extraction is a process of transforming the original features into a new set of features with more relevant and compact information. The purpose of the operation is to capture the essential information from the original features and represent it in the space of smaller features [4], [10].

Image segmentation is the process of partitioning a digital image into multiple image "segments", also known as image regions or objects (discrete groups of pixels) [11]. Through segmentation, the image is simplified, becoming easier to analyze and allowing the detection of limits in images (lines, curves) as well as the location of objects. Numerous algorithms have been developed for segmentation, most of them are oriented and combined with information specific to the field of applicability. In order to detect and track objects of interest in real time, image segmentation becomes extremely beneficial in video surveillance, including both people and vehicles. By applying image segmentation techniques, video surveillance systems can easily identify and isolate relevant objects, providing more accurate monitoring.

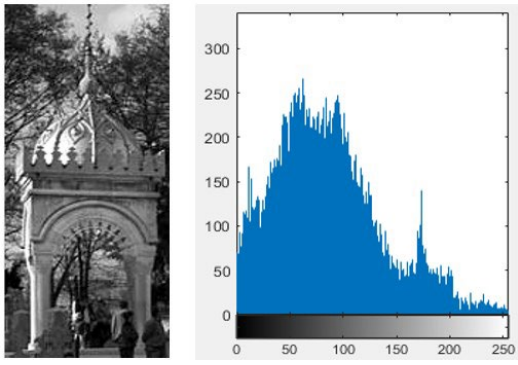

Fig. 1 – Original image and his histogram

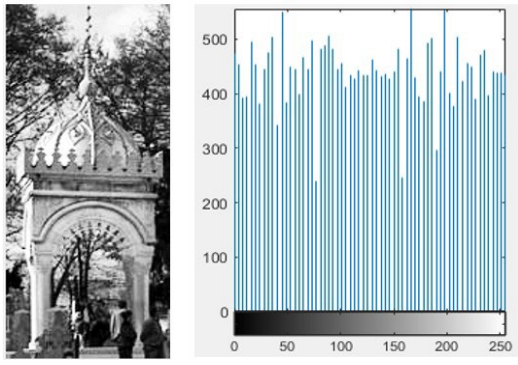

Fig. 2 – Image after histogram equalization

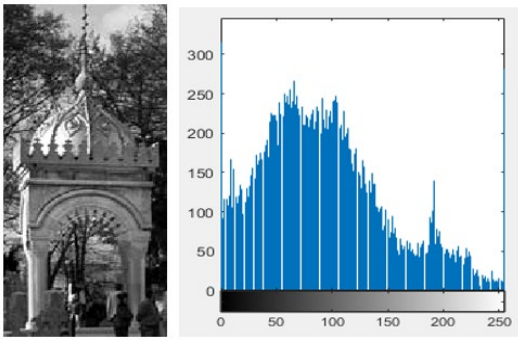

Fig. 3 – Adjusted image and his histogram

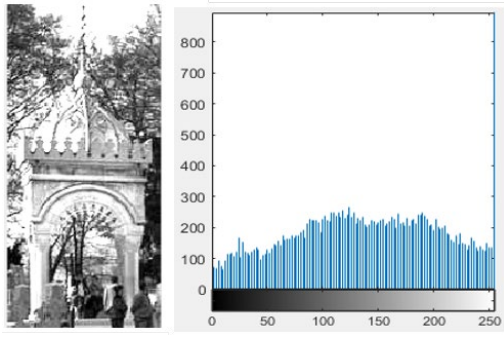

Fig. 4 – Adjusted image and his histogram

Image filtering has brought an evolutionary change in the field of image processing. Purpose: reduce noise, intensify the image, and identify features. Filtering is applied locally to each pixel in the image by replacing the intensity or color value of the current pixel with a value that depends on the intensity or color values of neighboring pixels (filter window). The number of neighbors considered determines the size of the filter. A filter can be defined as a matrix applied to each pixel and its adjacent neighbors in the given image [12]. This array is called a convolution kernel and operates on the image by applying convolutions. If I is an image defined as $I = [a_{i,j}]_{i=1:n,j=1:m}$, then each element a_{ij} (except the marginal elements), has 8 other elements around it, forming a 3x3 matrix. If B is a 3x3 mask matrix, the element c_{ij} of the convolution matrix $I \cdot B$ is obtained as follows:

$$c_{ij} = b_{ij}(a_{i-1,j-1} + a_{i-1,j} + a_{i-1,j+1} + a_{i,j-1} + a_{i,j} + a_{i,j+1} + a_{i+1,j-1} + a_{i+1,j} + a_{i+1,j+1})$$

$$i \neq 1, i \neq n, j \neq 1, j \neq m; c(1{:}n, 1) = 0; c(1{:}n, m) = 0;$$
$$c(1{,}1{:}m) = 0; c(n, 1{:}m) = 0$$

Convolutional kernels and filters are the building blocks of many computer vision applications [12]. More advanced algorithms and the combination of several types of convolutional kernels can lead to remarkable results in the detection and extraction of features from images.

The numerical values for Gauss, Laplace and LoG filters are obtained by known formulas:

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} = f(x+1, y) + f(x-1, y) + f(x, y+1) + f(x, y-1) - 4f(x, y)$$

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}, LoG(x, y) = -\frac{1}{\pi\sigma^4}\left[1 - \frac{x^2+y^2}{2\sigma^2}\right] e^{-\frac{x^2+y^2}{2\sigma^2}}$$

The filters used in the experiment are common image processing kernels: PREWITT, Sobel, Gaussian, Laplacian, Average, Log defined as follows.

Table 3 – The value of the filters used in the experiment

Prewitt filter	$\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$
Sobel Filter	$\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$
Laplace filter	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$
Average m*n filter	$\frac{1}{m*n}\begin{bmatrix} 1 & 1 & \cdots & 1 \\ \vdots & & \vdots & \vdots \\ 1 & 1 & \cdots & 1 \end{bmatrix}$

Gaussian filter approximation $x = -1{:}1{:}1;$ $y = x;\ \sigma = 1$	$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$
Gaussian filter $\sigma = 0.5;$ $x = -1{:}1{:}1;$ $y = x$	$\begin{bmatrix} 0.0113 & 0.0838 & 0.0113 \\ 0.0838 & 0.6193 & 0.0838 \\ 0.0113 & 0.0838 & 0.0113 \end{bmatrix}$
Laplacian filter	$\begin{bmatrix} 0.1667 & 0.6667 & 0.1667 \\ 0.6667 & -3.333 & 0.6667 \\ 0.1667 & 0.6667 & 0.1667 \end{bmatrix}$
LoG filter	$\begin{bmatrix} 0.2835 & 0.6629 & 0.2835 \\ 0.6629 & -4.9006 & 0.6629 \\ 0.2835 & 0.6629 & 0.2835 \end{bmatrix}$

Fig. 5 – Original image and filtered image with Prewitt, Sobel, Gauss, Laplacian and Log filters

Edge detection and image segmentation are important applications of gradient orientation and magnitude. The magnitude of the gradient is used to identify regions with significant changes in intensity, and the orientation of the gradient gives the direction of the edge [5], [13]. Image segmentation involves dividing the image into regions that are identified by similarity of orientation and magnitude of the gradient [14-15].

Popular algorithms for detecting and matching features

Harris Corner Detection algorithm identifies corners in an image based on changes in intensity in different directions. Harris corner detection is widely used for feature-based image detection and alignment [1-2], [9]. FAST (Features from Accelerated Segment Test) is a real-time corner detection algorithm [2],[9]. SIFT (Scale-Invariant Feature Transform) detects points of interest at multiple scales and orientations and provides a robust descriptor for each point of interest, making it invariant to changes in scale, rotation, and lighting [2]. SURF (Speeded-Up Robust Features) is an effective alternative to SIFT. It uses integral images to accelerate computation and provides similar performance in terms of robustness and invariance in scale and rotation [10]. BRIEF, which stands for Binary Robust Independent Elementary Features, is a feature descriptor algorithm used in computer vision for image processing tasks like object recognition and image matching. ORB (Oriented FAST and Rotated BRIEF) is a fusion between FAST corner detection and the BRIEF descriptor. It also aims to provide an efficient and real-time alternative to SIFT and SURF. ORB captures the characteristics of objects at different scales and orientations. FAST identifies points of interest by comparing the brightness of a central pixel to the surrounding 16 pixels, and whether more than 8 of these surrounding pixels are either darker or brighter than the centre pixel is considered a key point. BRISK (Binary Robust Invariant Scalable Keypoints) is a feature detection and description algorithm. BRISK is modular. This feature allows it to be combined with other methods or algorithms for detecting and extracting features [2]. MSER (Maximally Stable Extremal Regions) identifies regions in an image where significant intensity level changes occur [2]. MinEigen is used in feature extraction, image recording, and object recognition. MinEigen tests each pixel in an image to determine if it corresponds to a corner. It considers a small area centred around the pixel and calculates the minimum eigenvalue of the structure tensor in that area [2].

3. EXPERIMENTAL RESULTS AND DISCUSSIONS

The purpose of the experiment is to analyze and compare the results regarding the detection of objects in images using several detection methods. Four original and filtered images were used for both objects and scenes.

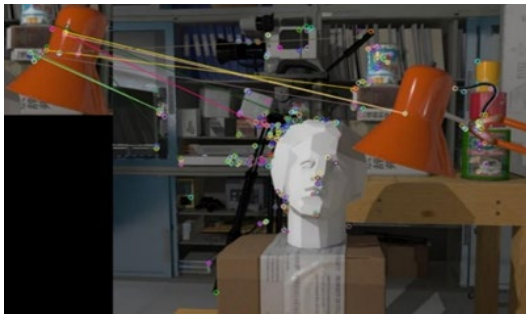

Fig. 6 – Image 1 with object 1 [16]

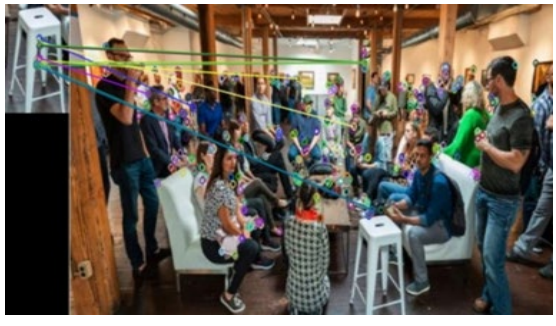

Fig. 7 – Image 2 with object 2 [17]

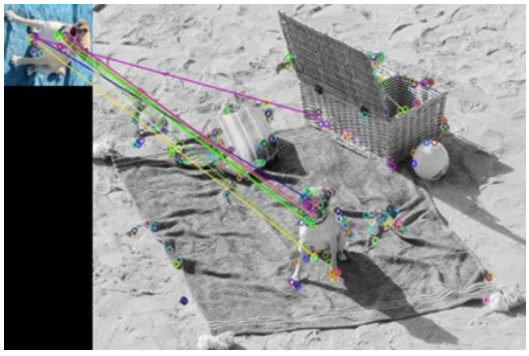

Fig. 8 – Image 3 with object 3 [18]

Fig. 9 – Image 4 with object 4

Below are the histograms of the four objects and scenes of each image.

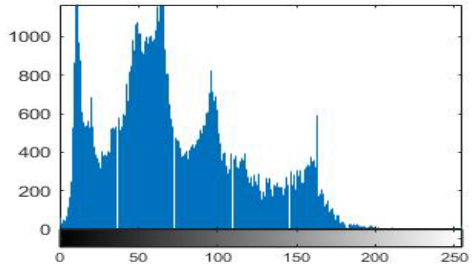

Fig. 10 – Intensity image 1 (object 1 histogram)

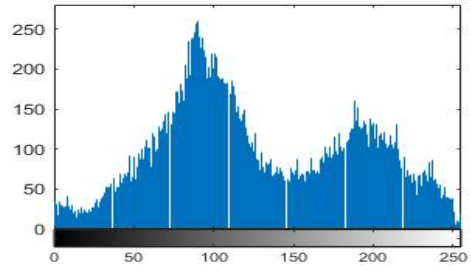

Fig. 11 – Intensity image 2 (object 2 histogram)

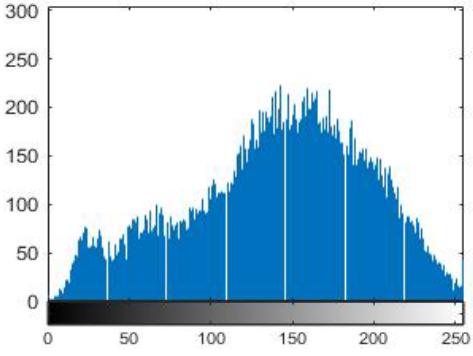

Fig. 12 – Intensity image 3 (object 3 histogram)

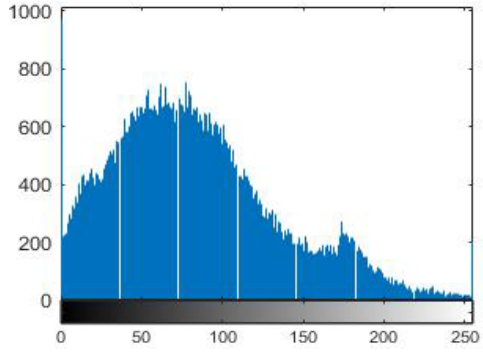

Fig. 13 – Intensity image 4 (object 4 histogram)

The information provided in Table 4 represent the performance of various feature detection methods (Harris, MinEigen, SURF, ORB, BRISK and MSER) when applied to images processed with different filters (I-orig, PREWITT, Sobel, Gaussian, Laplacian and LOG). The values in the tables represent the number of detected features from the original and filtered images by each method under the given filter conditions.

Table 4 – The number of features detected in original and filtered images using different algorithms

	Method\filters	I-orig	PREWITT	Sobel	Gaussian	Laplacian	LOG
Image 1	Harris	62	35	46	64	137	131
	MinEigen	160	132	147	156	309	301
	SURF	16	12	20	15	0	0
	ORB	43	60	82	41	28	91

	BRISK	30	76	128	23	3	54
	MSER	65	101	146	61	2	32
Image 2	Method\filters	I-orig	PREWITT	Sobel	Gaussian	Laplacian	LOG
	Harris	68	50	54	61	66	69
	MinEigen	133	111	116	124	115	122
	SURF	36	35	44	33	3	23
	ORB	14	17	23	13	27	38
	BRISK	91	101	142	79	77	153
	MSER	182	165	222	185	39	130
Image 3	Method\filters	I-orig	PREWITT	Sobel	Gaussian	Laplacian	LOG
	Harris	97	87	104	95	70	96
	MinEigen	147	131	141	156	148	155
	SURF	29	39	52	27	2	13
	ORB	32	29	37	23	41	48
	BRISK	156	226	280	119	175	312
	MSER	167	250	273	177	42	110
Image 4	Method\filters	I-orig	PREWITT	Sobel	Gaussian	Laplacian	LOG
	Harris	265	287	338	246	247	305
	MinEigen	513	477	503	529	485	516
	SURF	100	154	231	93	17	91
	ORB	800	708	808	636	859	1042
	BRISK	476	813	1106	315	653	1128
	MSER	182	371	507	167	104	262

From Table 4 we see that Harris performs consistently across all filters, with relatively stable feature detection numbers. However, its performance varies depending on the image. In Image 1, Harris detects more features with Laplacian and LOG filters, indicating its sensitivity to these filters. In Image 5, Harris detects the highest number of features with the Sobel filter.

MinEigen generally detects the most features across all methods and filters, especially under the Gaussian, Laplacian, and LOG filters. This suggests that MinEigen is highly sensitive to image features that are enhanced by these filters. SURF tends to detect fewer features compared to other methods, especially under the Laplacian and LOG filters, where it detects almost no features. This suggests that SURF may not perform well with images processed by these filters, possibly due to the nature of the keypoints it identifies. ORB's performance is varied, detecting more features with Sobel and LOG filters in some figures (e.g., Image 1 and Image 4). ORB shows significant variability depending on the filter applied, which might indicate that ORB is more sensitive to certain types of image transformations.

BRISK detects a high number of features with the Sobel and LOG filters, especially in Image 3 and Image 4. This suggests that BRISK is effective in identifying features in images with pronounced edges and noise, as Sobel and LOG often enhance these aspects. Transposed graphically, the top results lead to the observation that the efficiency of detection algorithms

is significantly dependent on the type of filter used. The results in the tables above are transposed into the graphs below.

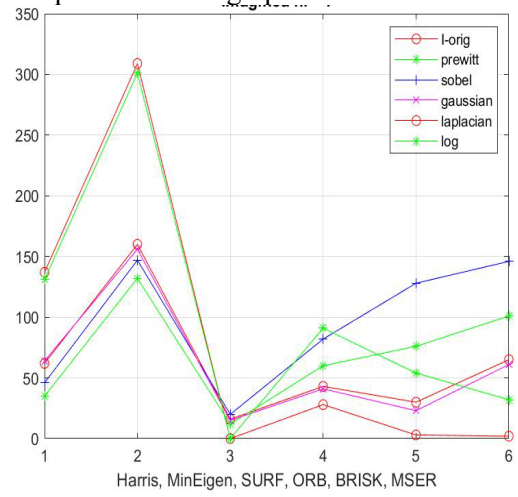

Fig. 14 – The number of features detected in Image 1

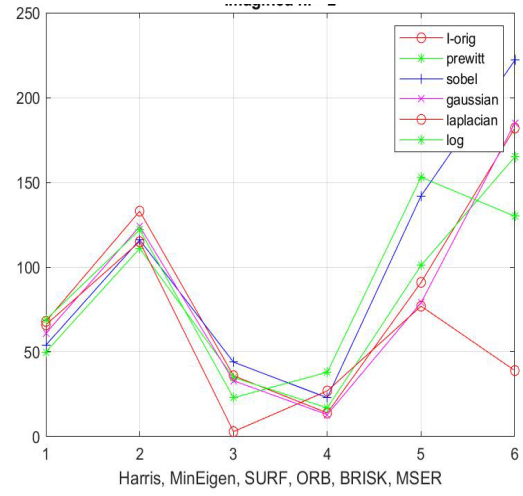

Fig. 15 – The number of features detected in Image 2

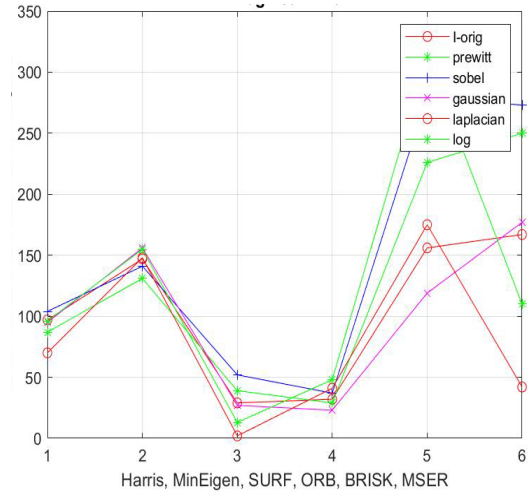

Fig. 16 – The number of features detected in Image 3

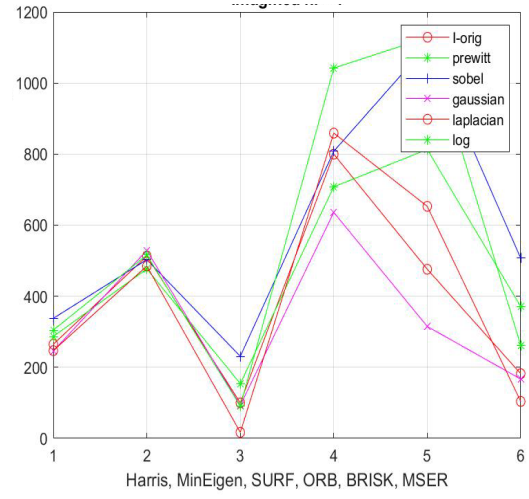

Fig. 17 – The number of features detected in Image 4

Analyzing the results in the table and graphs above, it can be seen that the efficiency of the detection algorithms is significantly dependent on the type of filter used.

Table 5 – Comparison of feature detection methods

Scene	Method	Object pts. Count	Scene pts. Count	No Pairs
1	SURF	16	424	15
	BRISK	7	259	2
	ORB	43	3265	25
	ALMOST	3	85	2
	MinEigen	71	1426	43

Scene	Method	Object pts. Count	Scene pts. Count	No Pairs
3	SURF	29	601	24
	BRISK	39	1059	15
	ORB	32	9850	19
	ALMOST	36	609	20
	MinEigen	44	2853	23

	MSER	65	308	11
Scene	Method	Object pts. Count	Scene pts. Count	No Pairs
2	SURF	36	926	26
	BRISK	26	1923	14
	ORB	14	9043	9
	ALMOST	16	1092	12
	MinEigen	41	1980	19
	MSER	182	1200	29

	MSER	167	839	28
Scene	Method	Object pts. Count	Scene pts. Count	No Pairs
4	SURF	100	1555	77
	BRISK	259	3737	59
	ORB	800	28373	415
	ALMOST	204	2549	117
	MinEigen	292	4767	94
	MSER	182	1569	39

Common Object Scene Features Detected with the MinEigen Algorithm

Fig. 18 – Matched points object scene using MinEigen algorithm

Fig. 19 – Matched points object scene using MinEigen algorithm

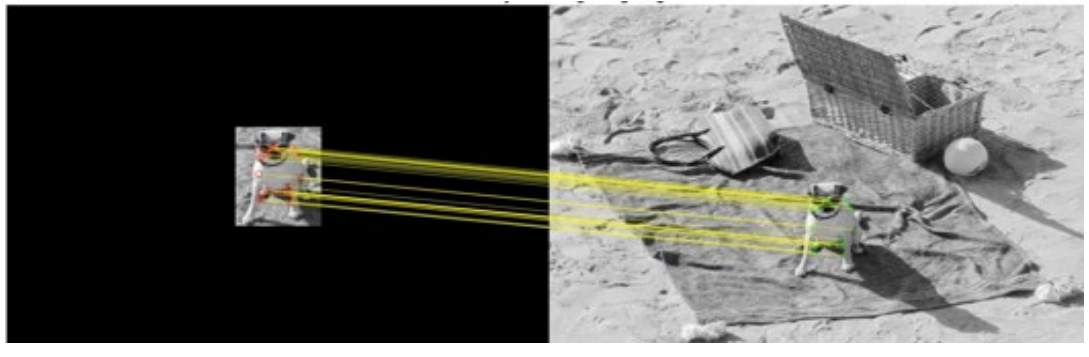

Fig. 20 – Matched points object scene using MinEigen algorithm

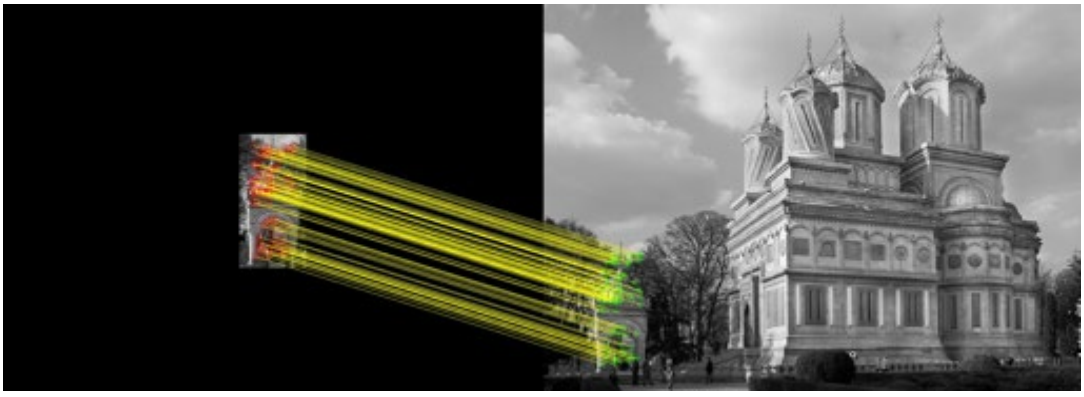

Fig. 21 – Matched points object scene using MinEigen algorithm

In the previous example it was noticed that the "bell" object was not detected in the scene containing it using the FAST and BRISK methods, so we proceeded to improve the image by applying a set of filters.

The result is shown in the table below. Since at least three pairs of object-scene points are required for detection, it can be seen that only the PREWITT, Sobel, and Sobel-Average filters determined at least three matching point perches.

Table 6 – Feature detection performance comparison of FAST and BRISK methods

FAST	Object pts. Count	Scene pts. Count	No Pairs	BRISK	Object pts. Count	Scene pts. Count	No Pairs
Orig Image 1 object 1	3	85	2	Orig Image 1 object 1	7	259	2
Filtered Image with				Filtered Image with			
PREWITT	22	427	11	PREWITT	31	752	5
Sobel	41	648	19	Sobel	53	1104	11
Gaussian	1	53	1	Gaussian	5	216	1
Laplacian	0	40	0	Laplacian	0	41	0
Average	1	12	1	Average	5	129	1
Log	20	372	0	Log	21	418	1
Log & Sobel	92	2047	3	Log & Sobel	111	2596	2
Sobel & Log	83	1774	9	Sobel & Log	104	2216	2
PREWITT & Laplacian	22	468	2	PREWITT & Laplacian	26	517	1
Sobel & Gaussian	31	540	18	Sobel & Gaussian	38	940	7
Sobel & Average	13	216	9	Sobel & Average	14	494	5
PREWITT & Average	4	87	2	PREWITT & Average	8	247	1

By applying a function to increase the contrast of object and scene images (mapping the values of the original pixels to new values), the object is detected in the scene even without filtering the images.

The effect of the filters can be analyzed from the table below.

Table 7 – Impact of contrast enhancement and filtering on feature detection using ALMOST and BRISK methods

FAST	Object pts. Count	Scene pts. Count	No Pairs	BRISK	Object pts. Count	Scene pts. Count	No Pairs
Imadjust (Image: bell)	25	194	5	Imadjust (Image: bell)	28	556	7
Filtered Image with				Filtered Image with			
Gaussian	1	53	1	Gaussian	17	466	4
Laplacian	0	40	0	Laplacian	21	141	0
Average	6	39	2	Average	11	306	4
Log	71	725	3	Log	21	418	1
Sobel & Log	153	2328	8	Sobel & Log	208	3013	5

Table 7 presents the number of object points detected, scene points detected and matching pairs identified by the ALMOST and BRISK feature detection methods after applying a contrast enhancement function (Imadjust [19]) and various image filters to the object and scene images. The results highlight how contrast enhancement alone can aid in object detection and how different filters further influence the detection and matching effectiveness of each method.

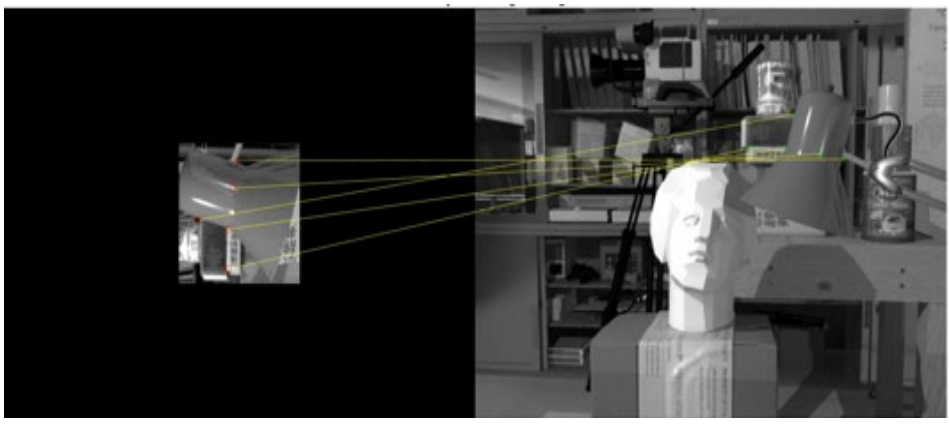

Fig. 22 – Matched points object scene using FAST algorithm

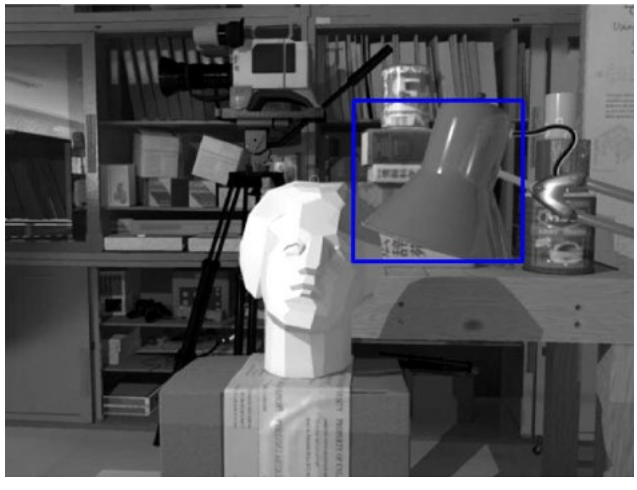

Fig. 23 – Detected object using FAST algorithm

To detect objects in images, algorithms also use distances between characteristic points determined in the object matrix and the scene matrix. It was previously shown that using the FAST method, the object Image1 ('bell') was detected in Scene1 using the following 5 pairs of points:

$$[xo_1, yo_1] = [80,25]; [xo_2, yo_2] = [74,57]; [xo_3, yo_3] = [28,94];$$
$$[xo_4, yo_4] = [68,105]; \ [xo_5, yo_5] = [67,152]$$

$$[xs_1, ys_1] = [501,177]; [xs_2, ys_2] = [469,171]; [xs_3, ys_3] = [432,125];$$
$$[xs_4, ys_4] = [420,165]; [xs_5, ys_5] = [374,164]$$

The pixel values corresponding to the dot pairs are:

Pixel_value_object = [13 233 4 203 246];

Pixel_value_scene = [17 185 20 170 185];

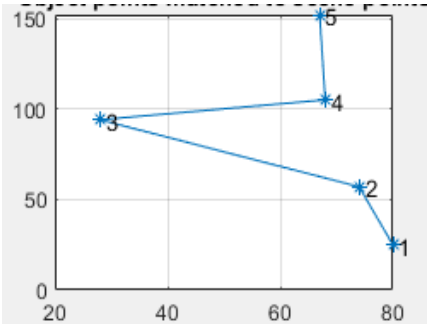

Fig. 24 – Object points matched to scene points

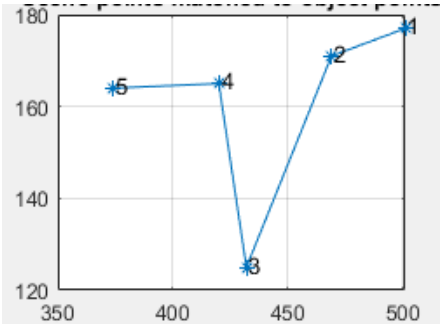

Fig. 25 – Scene points matched to object points

Table 8 shows the Euclidean distances between each pair of keypoints in the object image. Similarly, Table 9 shows the Euclidean distances between each pair of keypoints in the scene image. This table helps in understanding the spatial relationships between keypoints in the scene image and can be compared to the distances in the object image (Table 8) to analyze how the object has been transformed or deformed in the scene.

Table 8 – Object point distance

From point to point	1	2	3	4	5
1	0	32.55	86.40	80.89	127.66
2	32.55	0	59.03	48.37	95.25
3	86.40	59.03	0	41.48	69.89
4	80.89	48.37	41.48	0	47.01
5	127.66	95.25	69.89	47.01	0

Table 9 – Scene point distance

From point to point	1	2	3	4	5
1	0	32.55	86.40	81.29	127.66
2	32.55	0	59.03	49.87	95.45
3	86.40	59.03	0	41.48	69.89
4	80.89	48.37	41.48	0	47.01
5	127.66	95.25	69.89	47.01	0

4. THE EFFECT OF SCALING ON OBJECT DETECTION IN THE IMAGE

Initial Scenario: Detecting the Object at Scale 1

In the first scenario, the "bell" object was detected, extracted from a scene at scale 1, using the BRISK method.
After running the detection program, a total of 4 matched points between the object and the scene were identified. These matched points are shown in the table below:

Table 10 – Matched points between object and scene at Scale 1

Matched Points on Object	Matched Points on Scene
[80.5607, 24.5235]	[501.4812, 177.5984]
[27.6842, 145.8738]	[379.6916, 124.5747]
[68.4013, 107.4979]	[417.9175, 165.7861]
[86.3781, 124.9668]	[401.0456, 183.6526]

To assess how well the points on the object match those in the scene, the distances between the corresponding points were calculated.

The results show that the distances are very similar, indicating a correct and robust match between the object and the scene in the image.

Table 11 – Distances between object points at scale 1

From point to point	1	2	3	4
1	0	132.098	83.863	100.180
2	132.098	0	55.902	62.626
3	83.863	55.902	0	24.759
4	100.180	62.626	24.759	0

Table 12 – Distances between scene points at scale 1

From point to point	1	2	3	4
1	0	133.015	84.853	100.18
2	133.015	0	55.902	62.968
3	84.853	55.902	0	24.083
4	100.180	62.968	24.083	0

Scaling Scenario: Detecting the Object Scaled to 1.9x

In the second scenario, the "bell" object was scaled by 1.9 times, and the detection process was repeated using the BRISK method.

After running the program, 7 matched points between the scaled object and the scene were identified. The matched points and corresponding distances are shown in the table below.

Table 13 – Matched Points between the Object Scaled to 1.9x and the Scene

Matched Points on Object	Matched Points on Scene
[50.2424, 269.0007]	[384.5775, 123.1958]
[154.1524, 46.1317]	[501.4812, 177.5984]
[130.0989,200.2827]	[419.9166, 165.3867]
[52.9475, 275.9154]	[379.6916, 124.5747]

[130.3662, 202.3903]	[417.9175, 165.7861]
[52.1059, 276.8873]	[379.6916, 124.5747]
[138.0000, 162.0000]	[440.3186, 170.4785]

Table 14 – Distances between Object Points at Scale 1.9

From point to point	1	2	3	4	5	6	7
1	0	246.0589	105.6456	6.3246	104.3504	7.2801	138.5388
2	246.0589	0	155.8589	250.6891	157.8354	251.6029	117.0982
3	105.6456	155.8589	0	108.2081	2.0000	108.9036	38.8330
4	6.3246	250.6891	108.2081	0	106.8316	1.0000	142.0035
5	104.3504	157.8354	2.0000	106.8316	0	107.5174	40.7922
6	7.2801	251.6029	108.9036	1.0000	107.5174	0	142.8006
7	138.5388	117.0982	38.8330	142.0035	40.7922	142.8006	0

Table 15 – Distances between Scene Points at Scale 1.9

From point to point	1	2	3	4	5	6	7
1	0	128.8604	54.6717	5.0990	53.4135	5.0990	73.1095
2	128.860	0	82.8734	133.0150	84.8528	133.0150	61.4003
3	54.6717	82.8734	0	57.2800	2.0000	57.2800	21.5870
4	5.0990	133.0150	57.2800	0	55.9017	0	76.4003
5	53.4135	84.8528	2.0000	55.9017	0	55.9017	23.5372
6	5.0990	133.0150	57.2800	0	55.9017	0	76.4003
7	73.1095	61.40032	21.5870	76.4003	23.5372	76.4003	0

Analyzing the results obtained after the detection program of the scaled object we notice the following:

- The number of object points detected after scaling is no different from that without scaling (there are three almost identical pairs). Small distance between them.
- Coordinates of object points are multiplied by the scaling factor.

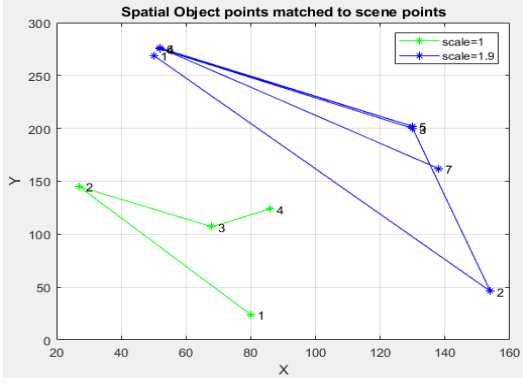

Fig. 26 – Spatial object points matched to scene points

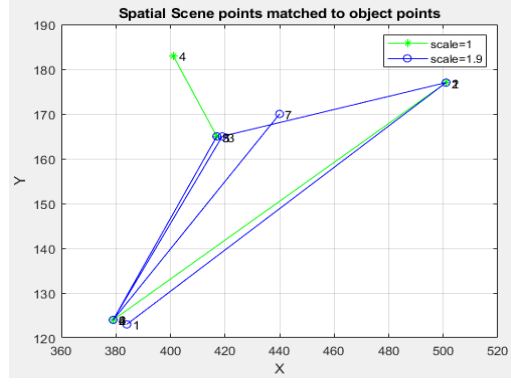

Fig. 27 – Spatial scene points matched to object points

For a good understanding of the role of gradients in the operation of determining pairs of similar object-scene points of interest, matrices of 21 x 21 pixels were built around a paired point, whose gradients were calculated.

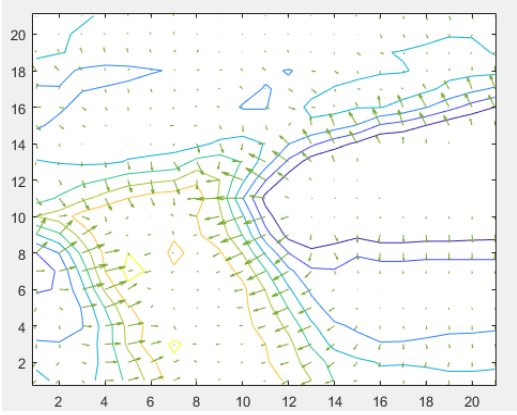

Fig. 28 – Iscontours and gradients in the 21x21 matrix around an object point rotated 90 degrees. The point is one of those suitable object-scenes

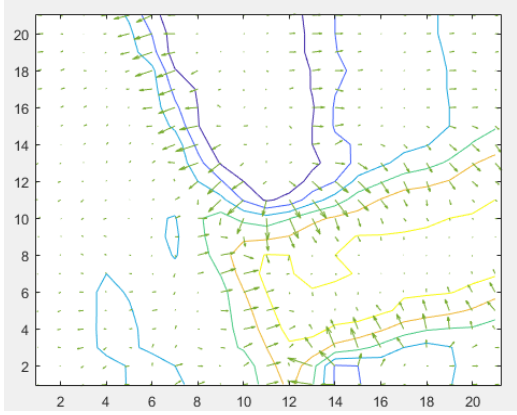

Fig. 29 – Isocontours and gradients in the 21x21 matrix around a scene point. The point is one of those suitable object-scenes

This study demonstrates the effectiveness of the BRISK method in detecting and matching features between objects and scenes, even under scaling transformations. The algorithm showed considerable robustness in identifying and matching keypoints, maintaining accuracy during significant scaling. This makes BRISK a solid choice for object detection applications where scale variations can be a major challenge.

5. CONCLUSIONS

The study demonstrates that preprocessing techniques, such as histogram equalization and pixel intensity adjustments, play a crucial role in enhancing feature detection. These preprocessing steps are necessary to improve the effectiveness of feature extraction algorithms like BRISK and SURF, particularly in scenarios where the original images do not provide sufficient detail for accurate object detection.

The results show that different feature detection algorithms exhibit varying levels of effectiveness depending on the image filters applied. For instance, the BRISK algorithm performs exceptionally well when combined with filters that enhance edges, such as Sobel and LOG, while SURF tends to detect fewer features under certain filtering conditions. This highlights the importance of selecting appropriate detection methods and filters based on the specific characteristics of the images being analyzed.

The experiment highlights the robustness of the BRISK algorithm in maintaining detection accuracy even under significant scaling transformations. The consistent matching of keypoints at both the original and scaled levels indicates that BRISK is a reliable choice for applications where objects may vary in size, such as in dynamic imaging environments.

The study also points out the limitations of certain algorithms, such as FAST and SURF, in detecting objects under specific conditions. For example, FAST struggled to detect objects without additional filtering, while SURF was less effective when applied to images processed with the Laplacian and LOG filters. These limitations suggest the need for a combined approach or algorithm selection tailored to the specific requirements of the application.

ACKNOWLEDGEMENT

This research is supported by INCAS – National Institute for Aerospace Research "Elie Carafoli", as a beneficiary of the NUCLEU Program, project code PN-23-17-06-03, Ctr. no. 36 N/12.01.2023, with the Ministry of Research, Innovation and Digitalization.

REFERENCES

[1] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, Speeded-Up Robust Features (SURF), *Computer Vision and Image Understanding (CVIU)*, vol. **110**, no. 3, pp. 346-359, 2008.
[2] A. Mohapatra, S. Sarangi, S. Patnaik, S. Sabut, Comparative study of corner and feature extractors for real-time object recognition in image processing, *Journal of information and communication convergence engineering*, vol.**12**, no. 4, pp. 263-270, 2014.
[3] B. Jähne, *Digital Image Processing; 6th revised and extended edition*, Springer Science & Business Media, 2005, ISBN 978-3-540-24035-8.
[4] K. Mikolajczyk, C. Schmid, A Performance Evaluation of Local Descriptors, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. **27**, no. 10, pp. 1615-1630, 2005.
[5] K. Mikolajczyk, T. Tuytelaars, Local Features in Images: A Survey, *Foundations and Trends in Computer Graphics and Vision*, vol. **3**, no. 3, pp. 177-280, 2007.
[6] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, 3rd Edition, Prentice Hall, 2008.
[7] S. Mallat, *A Wavelet Tour of Signal Processing*, 2nd Edition, Academic Press, 1999.
[8] K. P. Valavanis and G. J. Vachtsevanos, *Handbook of Unmanned Aerial Vehicles*, Springer, 2015.
[9] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, ORB: An Efficient Alternative to SIFT or SURF, in *Proceedings of the 2011 International Conference on Computer Vision (ICCV)*, 2011, pp. 2564-2571.
[10] D. G. Lowe, Distinctive Image Features from Scale-Invariant Keypoints, *International Journal of Computer Vision*, vol. **60**, no. 2, pp. 91-110, 2004.
[11] W. K. Pratt; *Digital Image Processing, Fourth Edition*, Wiley Interscience A John Wiley & Sons, Inc., Publication 2007.
[12] R. Szeliski, *Computer Vision: Algorithms and Applications*, Springer, London, 2010.
[13] B. K. P. Horn, *Robot Vision*, MIT Press, Cambridge, MA, 1986.
[14] C. Harris and M. Stephens, A Combined Corner and Edge Detector, in *Proceedings of the 4th Alvey Vision Conference*, 1988, pp. 147-151.
[15] E. Rosten and T. Drummond, Machine learning for high-speed corner detection, in *Proceedings of the 9th European Conference on Computer Vision (ECCV)*, 2006, pp. 430-443.
[16] * * * Matlab ImageDataStore, https://www.mathworks.com/help/vision/ug/monocular-visual-odometry.html
[17] * * * MeetFrank Platform, https://meetfrank.com/
[18] * * * https://www.samsung.com/
[19] * * * Matlab-Computer Vision System Toolbox™.