ANALOGY OF POINT FEATURE EXTRACTION TECHNIQUES

A SEMINAR REPORT

submitted by

VELAYUDHAN A	**- AM.EN.U4EEE17101**
NANDAKISHORE R NAIR	**- AM.EN.U4EEE17123**
NEERAJ S	**- AM.EN.U4EEE17124**
VISHNU S	**- AM.EN.U4EEE17147**

in partial fulfilment for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

ELECTRICAL AND ELECTRONICS ENGINEERING

AMRITA SCHOOL OF ENGINEERING, AMRITAPURI

AMRITA VISHWA VIDYAPEETHAM
AMRITAPURI 690 525
MAY 2020

AMRITA VISHWA VIDYAPEETHAM
AMRITA SCHOOL OF ENGINEERING, AMRITAPURI, 690525

AMRITA
VISHWA VIDYAPEETHAM

BONAFIDE CERTIFICATE

This is to certify that the seminar report entitled "**ANALOGY OF POINT FEATURE EXTRACTION TECHNIQUES**" submitted by

" **VELAYUDHAN A - AM.EN.U4EEE17101**
NANDAKISHORE R NAIR - AM.EN.U4EEE17123
NEERAJ S - AM.EN.U4EEE17124
VISHNU S - AM.EN.U4EEE17147

in partial fulfilment of the requirements for the award of the Degree, **Bachelor of Technology** in " **ELECTRICAL AND ELECTRONICS ENGINEERING** " is a bonafide record of the work carried out under my guidance and supervision at Amrita School of Engineering, Amritapuri.

Vivek A
Assistant professor
Dept. Electrical and Electronics Engineering

Dr Manjula G Nair
Professor & Chairperson
Dept. of Electrical and Electronics Engineering

ACKNOWLEDGEMENT

Presentation inspiration and motivation have always played a key role in the success of any venture.

We pay our deep sense of gratitude to Dr Manjula G Nair (HOD) Electrical and electronics department Amrita School of Engineering Amritapuri to encourage us to the highest peak and to provide us with the opportunity to prepare the project. We are immensely obliged to our friends for their elevating inspiration, encouraging guidance and kind supervision in the completion of our report.

We feel to acknowledge our indebtedness and a deep sense of gratitude to our guide Mr Vivek Asst Prof Department of Electrical and Electronics Engineering, valuable guidance and kind supervision is given to us throughout the course which shaped the present work as it shows.

Last but not least our parents are also an important inspiration for us. So with due regards, we express our gratitude to them.

FEATURE EXTRACTION

The human brain is capable of extracting, processing and manipulating data efficiently as information based on the neural pathways residing over our nervous system with the influence of neurons. However compared to the machine point of view this concept of identification is relatively harder as they are trained to perform over an algorithm for the fast computation. For example the computers can perform large amounts of calculation in milliseconds which would be a lifetime process for human brains. In contrast to that the humans can identify, differentiate, categorise the world around it which is harder for the machine's perspective. The artificial intelligence concept where the computers think like humans is the near future, but for the computer vision to attain the capability to identify, differentiate, categorise the world, it has to be trained under supervised or unsupervised learning. Image processing with feature extraction is an inevitable part of the machine training process and it is essential to find the most efficient feature extraction algorithm for every possible scenario.

Brisk Features

The BRISK(Binary Robust Invariant Scalable Keypoints) algorithm is a feature point detection and description algorithm with scale invariance and rotation invariance. It constructs the feature descriptor of the local image through the gray scale relationship of random point pairs in the neighborhood of the local image, and obtains the binary feature descriptor.The ideal keypoint descriptor captures the most important and distinctive information content enclosed in the detected salient regions, such that the same structure can be recognized if encountered.The factors like desired quality of keypoints, the speed of detection and description are used to distinguish different feature point extraction algorithms.

Among the conventional algorithms **Lowe's SIFT** approach is widely accepted as one of the highest quality options currently available, promising distinctiveness and invariance to a variety of common image transformations. However, the higher cost of computation is one of its drawback

-Another prominent algorithm is a combination of the FAST keypoint detector and the BRIEF approach to description which offers a much more suitable alternative for real-time applications.Although this has an advantage of lower computational requirement it suffers in terms of reliability and robustness as it has minimal tolerance to image distortions and transformations, in particular to in-plane rotation and scale change.

The Brisk algorithm can be explained in three steps namely Scale-Space Keypoint Detection, Keypoint Description and Descriptor Matching

Scale-Space Keypoint Detection

In this method image saliency is considered a continuous quantity not only across the image but also along the scale dimension, then perform a sub-pixel and continuous scale refinement for each detected maximum.

Keypoint Description

In BRISK, we identify the characteristic direction of each keypoint to allow for orientation-normalized descriptors and hence achieve rotation invariance which is key to general robustness.the brightness comparisons with the focus on maximizing descriptiveness is also considered in this phase.

Descriptor Matching

the respective operations reduce to a bitwise XOR followed by a bit count, which can both be computed very efficiently on today's architectures.

FAST

Features from accelerated segment test (FAST) is a corner detection method, which could be used to extract feature points and later used to track and map objects in many computer vision tasks. The FAST corner detector was originally developed by Edward Rosten and Tom Drummond and was published in 2006. The most promising advantage of the FAST corner detector is its computational efficiency. Moreover, when machine learning techniques are applied, superior performance in terms of computation time and resources can be realized. The FAST corner detector is very suitable for real-time video processing application because of this high-speed performance.

The algorithm is explained below:

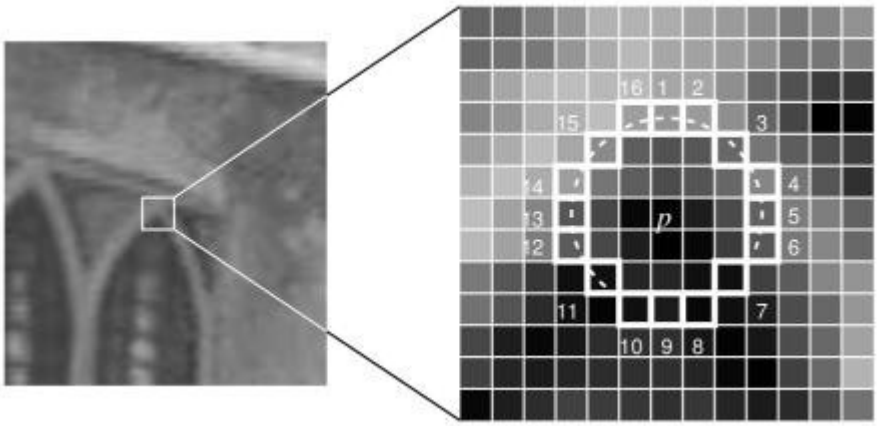


Fig 1. 12 point segment test corner detection in an image patch. The highlighted squares are the pixels used in the corner detection. The pixel at p is the center of a

candidate corner. The arc is indicated by the dashed line passes through 12 contiguous pixels which are brighter than p by more than the threshold.

The interest point detection in FAST can be explained in few steps as given below

- Select a pixel ***p*** in the image which is to be identified as an interest point or not. Let its intensity be ***Ip.***
- Select appropriate threshold value ***t***.
- Consider a circle of 16 pixels around the pixel under test.
- Now the pixel ***p*** is a corner if there exists a set of ***n*** contiguous pixels in the circle (of 16 pixels) which are all brighter than ***Ip + t***, or all darker than ***Ip - t***. (The authors have used ***n***= 12 in the first version of the algorithm)
- To make the algorithm fast, first compare the intensity of pixels 1, 5, 9 and 13 of the circle with ***Ip***. As evident from the figure above, at least three of these four pixels should satisfy the threshold criterion so that the interest point will exist.
- If at least three of the four-pixel values — ***I1, I5, I9, I13*** are not above or below ***Ip + t***, then ***p*** is not an interest point (corner). In this case reject the pixel ***p*** as a possible interest point. Else if at least three of the pixels are above or below ***Ip + t***, then check for all 16 pixels and check if 12 contiguous pixels fall in the criterion.
- Repeat the procedure for all the pixels in the image.

Machine Learning Approach

- Select a set of images for training (preferably from the target application domain)
- Run FAST algorithm in every image to find feature points.
- For every feature point, store the 16 pixels around it as a vector. Do it for all the images to get feature vector ***p***.
- Each pixel (say ***x***) in these 16 pixels can have one of the following three states:

$$S_{p\to x} = \begin{cases} d, & I_{p\to x} \le I_p - t & \text{(darker)} \\ s, & I_p - t < I_{p\to x} < I_p + t & \text{(similar)} \\ b, & I_p + t \le I_{p\to x} & \text{(brighter)} \end{cases}$$

- Depending on these states, the feature vector ***P*** is subdivided into 3 subsets ***Pd, Ps, Pb***.

- Define a variable ***Kp*** which is true if ***p*** is an interest point and false if ***p*** is not an interest point.
- Use the ID3 algorithm (decision tree classifier) to query each subset using the variable Kp for the knowledge about the true class.
- The ID3 algorithm works on the principle of entropy minimization. Query the 16 pixels in such a way that the true class is found (interest point or not) with the minimum number of queries. Or in other words, select the pixel x, which has the most information about the pixel ***p***. The entropy for the set ***P*** can be mathematically represented as:

$$H(P) = (c + \bar{c}) \log_2(c + \bar{c}) - c \log_2 c - \bar{c} \log_2 \bar{c}$$
$$\text{where} \quad c = \left|\{p | K_p \text{ is true}\}\right| \quad \text{(number of corners)}$$
$$\text{and} \quad \bar{c} = \left|\{p | K_p \text{ is false}\}\right| \quad \text{(number of non corners)}$$

- This is recursively applied to all the subsets until its entropy is zero.
- The decision tree so created is used for fast detection in other images.

Non-maximal Suppression

Detecting multiple interest points in adjacent locations is another issue of the former approach. This issue is solved by using Non-maximum Suppression.

- Compute a score function, ***V*** for all the detected feature points. ***V*** is the sum of absolute difference between ***p*** and 16 surrounding pixels values.
- Consider two adjacent keypoints and compute their ***V*** values.
- Discard the one with lower ***V*** value.

Limitations of the FAST Algorithm

Other corner detection methods work very differently from the FAST method and a surprising result is that FAST does not detect corners on computer-generated images that are perfectly aligned to the ***x-axes*** and ***y-axes***. Since the detected corner must have a ring of darker or lighter pixel values around the center that includes both edges of the corner, crisp images do not work well.

This is because the FAST algorithm requires a ring of contrasting pixels more than three-quarters around the center of the corner. In the computer-generated image, both edges of a box at a corner are in the ring of the pixel used, so the test for a corner fails. A workaround to this problem is to add blur to the image so that the corners are less precise but can be detected.

HARRIS Corner Detection

Harris Corner Detector is a corner detection operator that is commonly used in computer vision algorithms to extract corners and infer features of an image. It was

first introduced by Chris Harris and Mike Stephens in 1988 upon the improvement of Moravec's corner detector. Compared to the previous one, Harris' corner detector takes the differential of the corner score into account with reference to direction directly, instead of using shifting patches for every 45-degree angle, and has been proved to be more accurate in distinguishing between edges and corners. Since then, it has been improved and adopted in many algorithms to preprocess images for subsequent applications.

The idea is to consider a small window around each pixel p in an image. We want to identify all such pixel windows that are unique. Uniqueness can be measured by shifting each window by a small amount in a given direction and measuring the amount of change that occurs in the pixel values.

More formally, we take the sum squared difference (SSD) of the pixel values before and after the shift and identify pixel windows where the SSD is large for shifts in all 8 directions. Let us define the change function E(u,v) as the **sum** of all the sum squared differences (SSD), where u,v are the x,y coordinates of every pixel in our 3 x 3 window and I is the intensity value of the pixel. The features in the image are all pixels that have large values of E(u,v), as defined by some threshold.

$$E(u,v) = \sum_{x,y} w(x,y)[I(x+u, y+v) - I(x,y)]^2$$

We have to maximize this function $E(u,v)$ for corner detection. That means, we have to maximize the second term. Applying Taylor Expansion to the above equation and using some mathematical steps, we get the final equation as:

$$E(u,v) \approx \begin{bmatrix} u & v \end{bmatrix} \left(\sum \begin{bmatrix} {I_x}^2 & I_x I_y \\ I_x I_y & {I_y}^2 \end{bmatrix} \right) \begin{bmatrix} u \\ v \end{bmatrix}$$

Now, we rename the summed-matrix, and put it to be M:

$$M = \sum w(x,y) \begin{bmatrix} {I_x}^2 & I_x I_y \\ I_x I_y & {I_y}^2 \end{bmatrix}$$

So the equation now becomes:

$$E(u,v) \approx \begin{bmatrix} u & v \end{bmatrix} M \begin{bmatrix} u \\ v \end{bmatrix}$$

Remember that we want the SSD to be large in shifts for all eight directions, or conversely, for the SSD to be small for none of the directions. By solving for the eigenvectors of M, we can obtain the directions for both the largest and smallest increases in SSD. The corresponding eigenvalues give us the actual value amount of

these increases. A score, R, is calculated for each window:

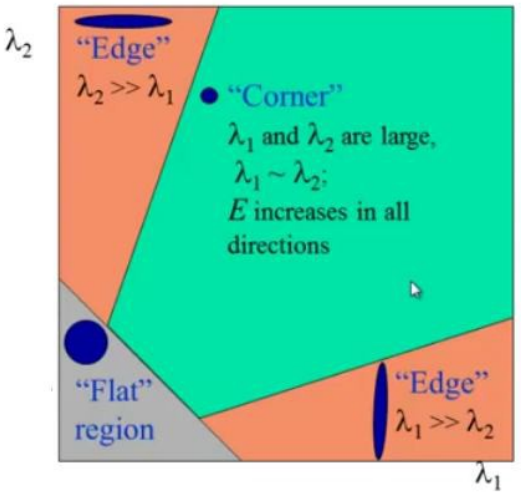
$$R = \det M - k(trace\, M)^2$$

$$\det M = \lambda_1 \lambda_2$$

$$trace\, M = \lambda_1 + \lambda_2$$

$\lambda 1$ and $\lambda 2$ are the eigenvalues of M. So the values of these eigenvalues decide whether a region is a corner, edge or flat.

- When $|R|$ is small, which happens when $\lambda 1$ and $\lambda 2$ are small, the region is flat.
- When $R<0$, which happens when $\lambda 1>>\lambda 2$ or vice versa, the region is an edge.
- When R is large, which happens when $\lambda 1$ and $\lambda 2$ are large and $\lambda 1 \sim \lambda 2$, the region is a corner.



High-level pseudocode

1. Take the grayscale of the original image

2. Apply a Gaussian filter to smooth out any noise

3. Apply Sobel operator to find the x and y gradient values for every pixel in the grayscale image

4. For each pixel p in the grayscale image, consider a 3×3 window around it and compute the corner strength function. Call this its Harris value.

5. Find all pixels that exceed a certain threshold and are the local maxima within a certain window (to prevent redundant dupes of features)

6. For each pixel that meets the criteria in 5, compute a feature descriptor.

KAZE

P. F. Alcantarilla et al. put forward KAZE features in 2012 that exploit non-linear scale space through non-linear diffusion filtering. This makes blurring in images locally adaptive to feature-points, thus reducing noise and simultaneously retaining the boundaries of regions in subject images. KAZE detector is based on scale normalized determinant of Hessian Matrix which is computed at multiple scale levels. The maxima of detector response are picked up as feature-points using a moving window. Feature description introduces the property of rotation invariance by finding dominant orientation in a circular neighborhood around each detected feature. KAZE features are invariant to rotation, scale, limited affine and have more distinctiveness at varying scales with the cost of moderate increase in computational time. The below equation shows the standard nonlinear diffusion formula.

$$\frac{\partial L}{\partial t} = div(c(x, y, t).\nabla L)$$

Where “c” is conductivity function, “div” is divergence, “▽” is gradient operator and “L” is image luminance.

MinEigenFeatures detection

The Shi-Tomasi corner detector is based entirely on the Harris corner detector. However, one slight variation in a "selection criteria" made this detector much better than the original. It works quite well where even the Harris corner detector fails. So here's the minor change that Shi and Tomasi did to the original Harris corner detector.

The Harris corner detector has a corner selection criteria. A score is calculated for each pixel, and if the score is above a certain value, the pixel is marked as a corner. The score is calculated using two eigenvalues. That is, you gave the two eigenvalues to a function. The function manipulates them, and gave back a score.

Shi and Tomasi suggested that the function should be done away with. Only the eigenvalues should be used to check if the pixel was a corner or not.

The score for **Harris** corner detector was calculated like this (R is the score):

$$R = \det M - k(trace\, M)^2$$

$$\det M = \lambda_1 \lambda_2$$

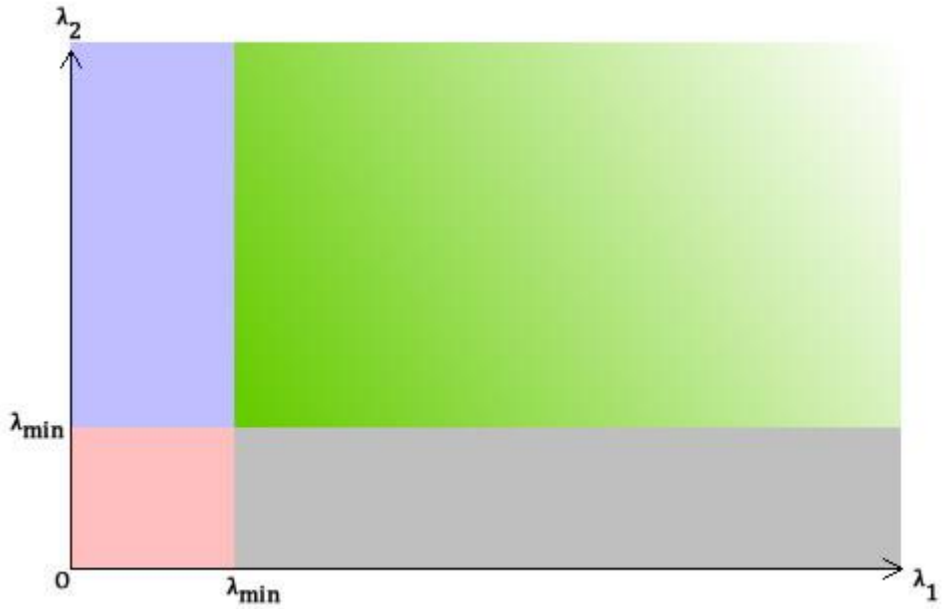
$$trace\, M = \lambda_1 + \lambda_2$$

For **Shi-Tomasi**, it's calculated like this:

$$R = min(\lambda_1, \lambda_2)$$

In their paper, Shi and Tomasi demonstrated experimentally that this score criteria was much better. If R is greater than a certain predefined value, it can be marked as a corner.

the effect region for a point to be a corner is something like this:



- Green: both λ1 and λ2 are greater than a certain value. Thus, this region is for pixels "accepted" as corners.
- In the blue and gray regions, either λ1 or λ2 is less than the required minimum.
- In the red region, both λ1 and λ2 are less than the required minimum. Compare the above with a similar graph for Harris corner detector... You'll see the blue and gray areas are equivalent to the "edge" areas. The red region is for "flat" areas. The green is for corners.

MSER feature extraction-Maximally Stable ExtremarMl Regions

MSER is a method for blob **detection** in images. The **MSER algorithm** extracts from an image a number of co-variant regions, called **MSERs**: an **MSER** is a stable connected component of some gray-level sets of the image .

MSER is based on the idea of taking regions which stay nearly the same through a wide range of thresholds. All the pixels below a given threshold are white and all those above or equal are black. If we are shown a sequence of thresholded images I with frame t corresponding to threshold t, we would first see a black image, then white spots corresponding to local intensity minima will appear then grow larger. These white spots will eventually merge, until the whole image is white. The set of all connected components in the sequence is the set of all extremal regions.

The MSER extraction implements the following steps:

– Sweep threshold of intensity from black to white, performing a simple luminance thresholding of the image

– Extract connected components (“Extremal Regions”)

– Find a threshold when an extremal region is “Maximally Stable”, i.e. local minimum of the relative growth of its square. Due to the discrete nature of the image, the region below / above may be coincident with the actual region, in which case the region is still deemed maximal.

– Approximate a region with an ellipse (this step is optional)

– Keep those regions descriptors as features

Original image MSER regions MSER ellipses

ORB (Oriented FAST and Rotated BRIEF)

Oriented FAST and Rotated BRIEF (ORB) was developed at OpenCV labs by Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary R. Bradski in 2011, as an efficient and viable alternative to SIFT and SURF. ORB was conceived mainly because SIFT and SURF are patented algorithms. ORB, however, is free to use.

Given a pixel p in an array fast compares the brightness of p to surrounding 16 pixels that are in a small circle around p. Pixels in the circle are then sorted into three classes (lighter than p, darker than p or similar to p). If more than 8 pixels are darker or brighter than p than it is selected as a keypoint. So keypoints found by fast gives us information of the location of determining edges in an image.

After locating keypoints orb now assign an orientation to each keypoint like left or right facing depending on how the levels of intensity change around that keypoint. For detecting intensity change orb uses intensity centroid. The intensity centroid assumes that a corner's intensity is offset from its center, and this vector may be used to impute an orientation.

First, the moments of a patch are defined as:

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

ORB descriptor-Patch's moment's definition

With these moments we can find the centroid, the "center of mass" of the patch as:

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

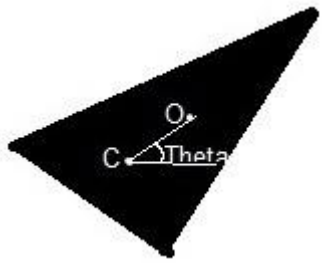
ORB descriptor — Center of the mass of the patch

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

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

ORB descriptor — Orientation of the patch

Here is an illustration to help explain the method:



Once we've calculated the orientation of the patch, we can rotate it to a canonical rotation and then compute the descriptor, thus obtaining some rotation invariance.

Brief takes all keypoints found by the fast algorithm and converts it into a binary feature vector so that together they can represent an object. Binary features vector also known as binary feature descriptor is a feature vector that only contains 1 and 0. In brief, each keypoint is described by a feature vector which is 128–512 bits string.

Brief start by smoothing the image using a Gaussian kernel in order to prevent the descriptor from being sensitive to high-frequency noise.Then brief select a random pair of pixels in a defined neighborhood around that keypoint. The defined neighborhood around a pixel is known as a patch, which is a square of some pixel width and height. The first pixel in the random pair is drawn from a Gaussian distribution centered around the keypoint with a stranded deviation or spread of sigma. The second pixel in the random pair is drawn from a Gaussian distribution centered around the first pixel with a standard deviation or spread of sigma by two. Now if the first pixel is brighter than the second, it assigns the value of 1 to

corresponding bit else 0.

Again brief select a random pair and assign the value to them. For a 128-bit vector, brief repeat this process for 128 times for a keypoint. Brief create a vector like this for each keypoint in an image. However, BRIEF also isn't invariant to rotation so orb uses rBRIEF(Rotation-aware BRIEF). ORB tries to add this functionality, without losing out on the speed aspect of BRIEF.

ORB specifies the rBRIEF algorithm as follows:

1) Run each test against all training patches.

2) Order the tests by their distance from a mean of 0.5, forming the vector T.

3) Greedy search:

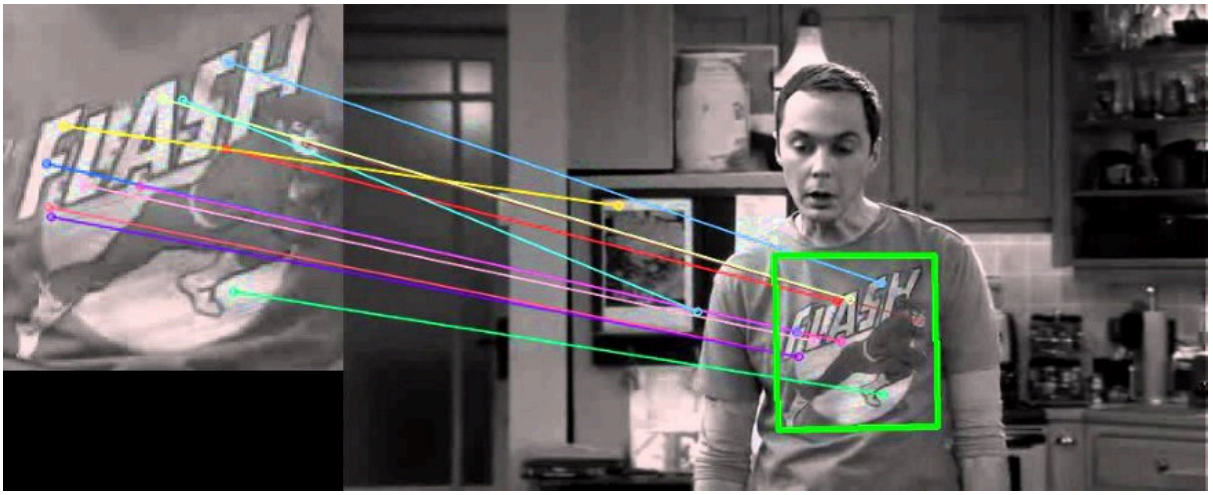
- Put the first test into the result vector R and remove it from T.
- Take the next test from T, and compare it against all tests in R. If its absolute correlation is greater than a threshold, discard it; else add it to R.

- Repeat the previous step until there are 256 tests in R. If there are fewer than 256, raise the threshold and try again

rBRIEF shows significant improvement in the variance and correlation over steered BRIEF.

SURF (Speeded-Up Robust Features)

The SURF method (Speeded Up Robust Features) is a fast and robust algorithm for local, similarity invariant representation and comparison of images. The main interest of the SURF approach lies in its fast computation of operators using box filters, thus enabling real-time applications such as tracking and object recognition.



SURF is composed of two steps

- **Feature Extraction**
- **Feature Description**

Feature Extraction

The approach for interest point detection uses a very basic Hessian matrix approximation.

Integral images

The Integral Image or *Summed-Area Table* was introduced in 1984. The Integral Image is used as a quick and effective way of calculating the sum of values (pixel values) in a given image or a rectangular subset of a grid (the given image). It is mainly used for calculating the average intensity within a given image.

They allow for fast computation of box type convolution filters. The entry of an integral image $I_{\sum}(x)$ at a location $x = (x,y)^T$ represents the sum of all pixels in the

input image I within a rectangular region formed by the origin and x.

$$I_{\Sigma}(\mathbf{x}) = \sum_{i=0}^{i \leqslant x} \sum_{j=0}^{j \leqslant y} I(i,j)$$

With I_Σ calculated, it only takes four additions to calculate the sum of the intensities over any upright, rectangular area, independent of its size.

Hessian matrix-based interest points

Surf uses the Hessian matrix because of its good performance in computation time and accuracy. Rather than using a different measure for selecting the location and the scale (Hessian-Laplace detector), surf relies on the **determinant of the Hessian matrix** for both. Given a pixel, the Hessian of this pixel is something like:

$$H(f(x,y)) = \begin{bmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} \\ \frac{\partial^2 f}{\partial x \partial y} & \frac{\partial^2 f}{\partial y^2} \end{bmatrix}$$

For adapt to any scale, we filtered the image by a Gaussian kernel, so given a point

X = (x, y), the Hessian matrix H(x, σ) in x at scale σ is defined as:

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

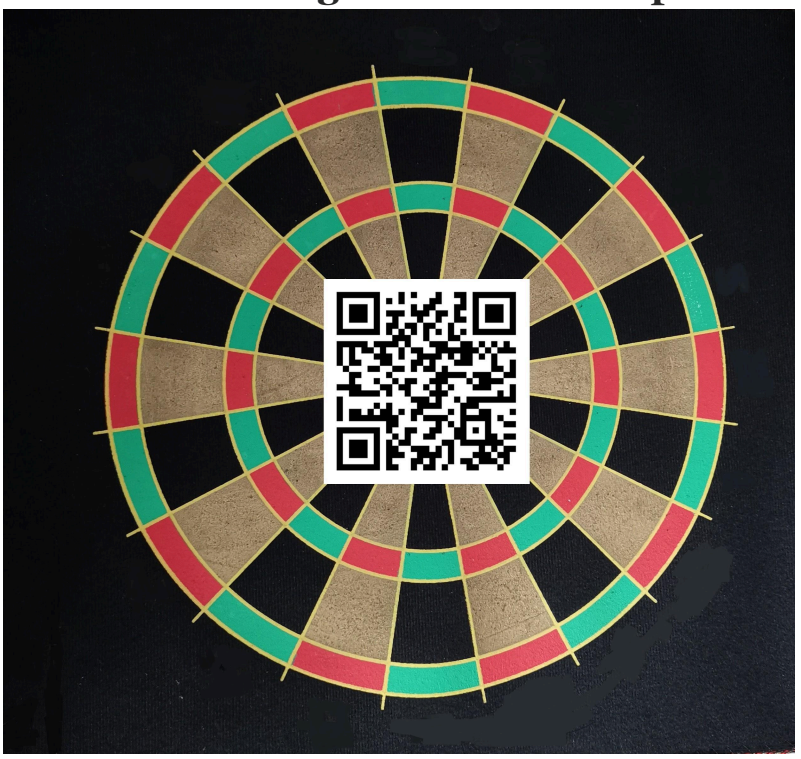
where Lxx(x, σ) is the convolution of the Gaussian second order derivative with the image I in point x, and similarly for Lxy (x, σ) and Lyy (x, σ).

Feature Description

The creation of SURF descriptor takes place in two steps. The first step consists of fixing a reproducible orientation based on information from a circular region around the keypoint. Then, we construct a square region aligned to the selected orientation and extract the SURF descriptor from it.

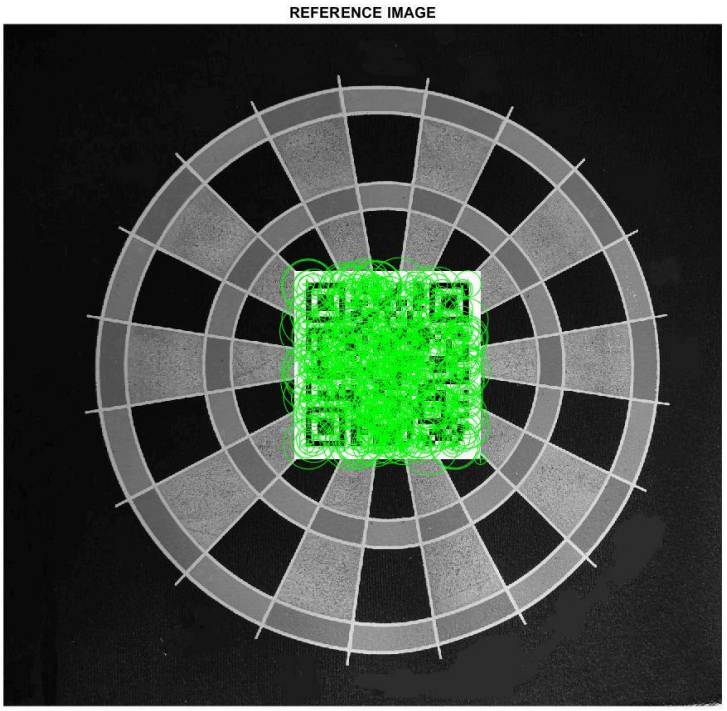
As we know, the factors like desired quality of keypoints, the speed of detection and description are used to distinguish different feature point extraction algorithms.Thus a visualisation of the data acquired on these factors is necessary for finding the best feature extraction algorithm for a given scenario.

The reference image used for the experiment

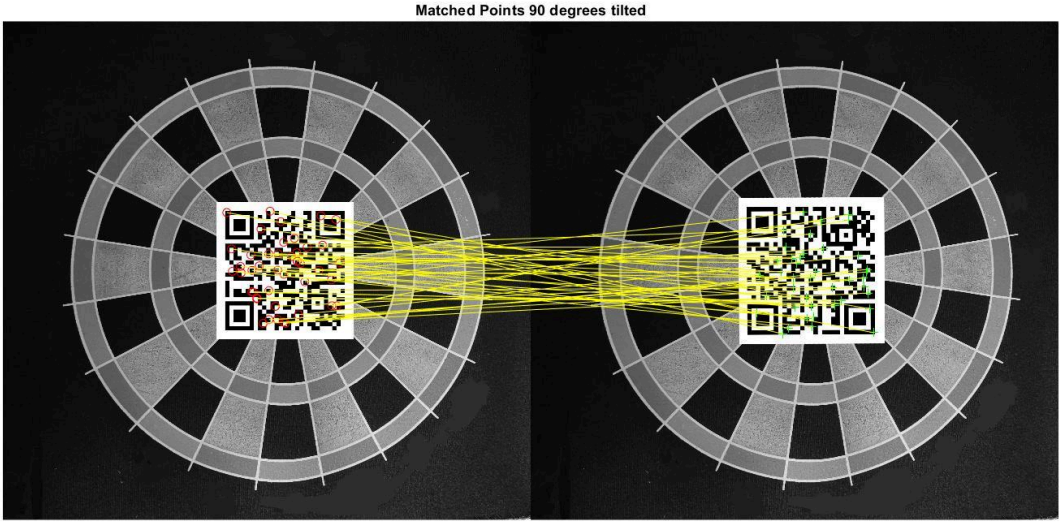


SIMULATION RESULTS

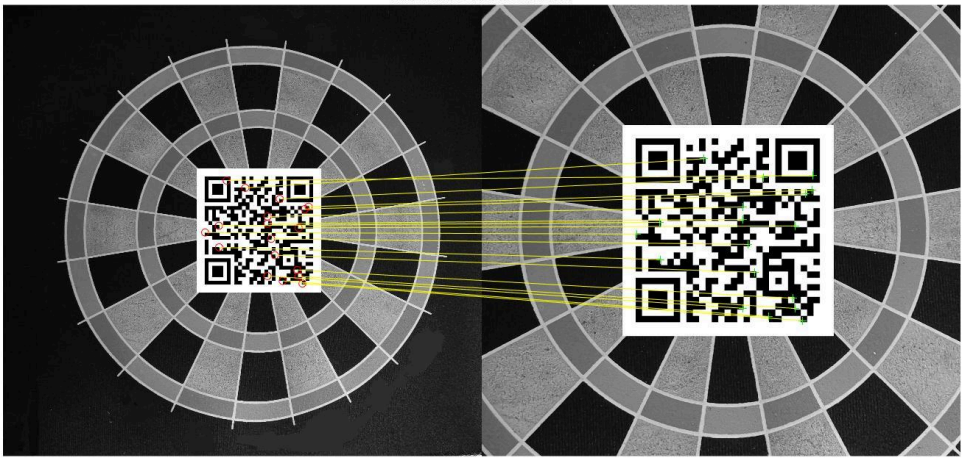
BRISK



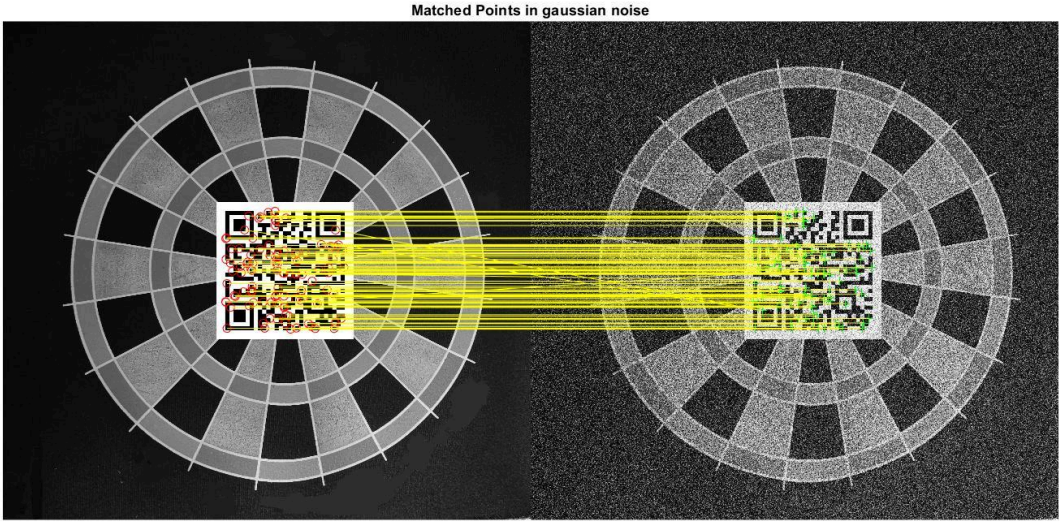
Matched Points 90 degrees tilted



Matched Points in scaled 1.7

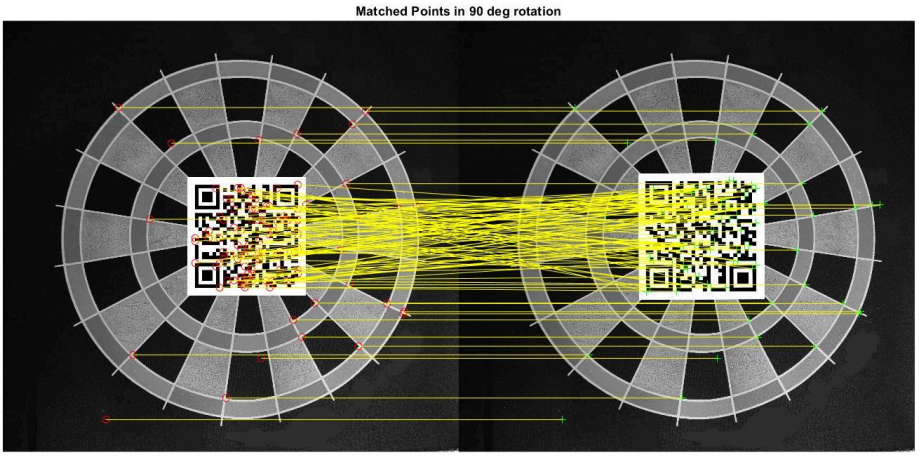


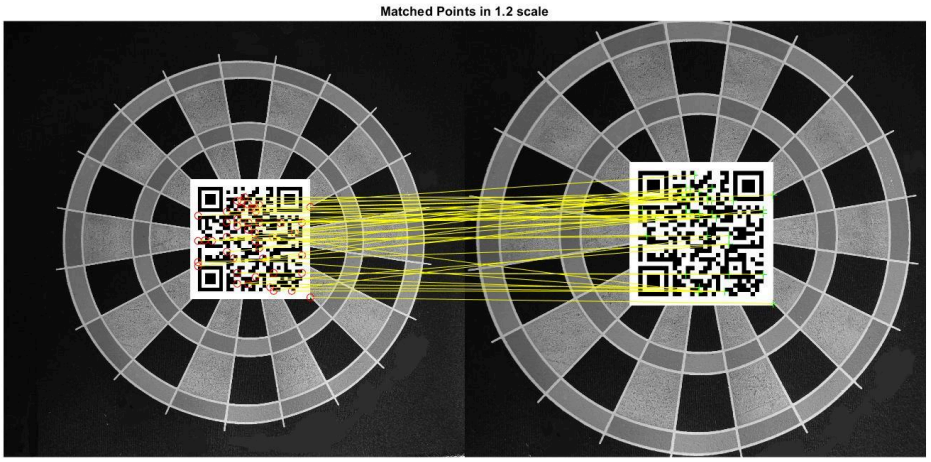
Matched Points in gaussian noise



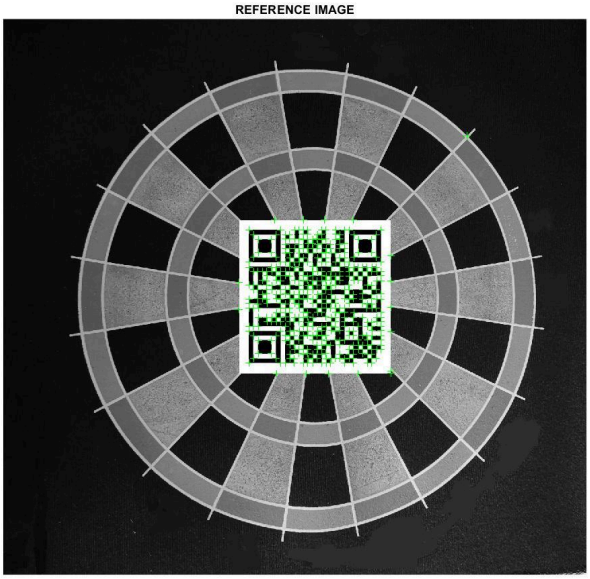
FAST

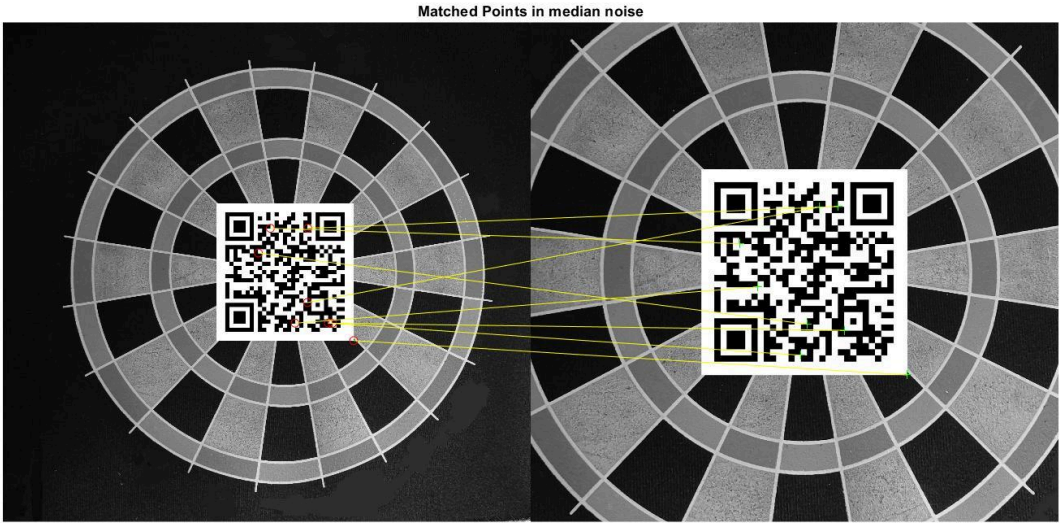

REFERENCE IMAGE


Matched Points in 90 deg rotation

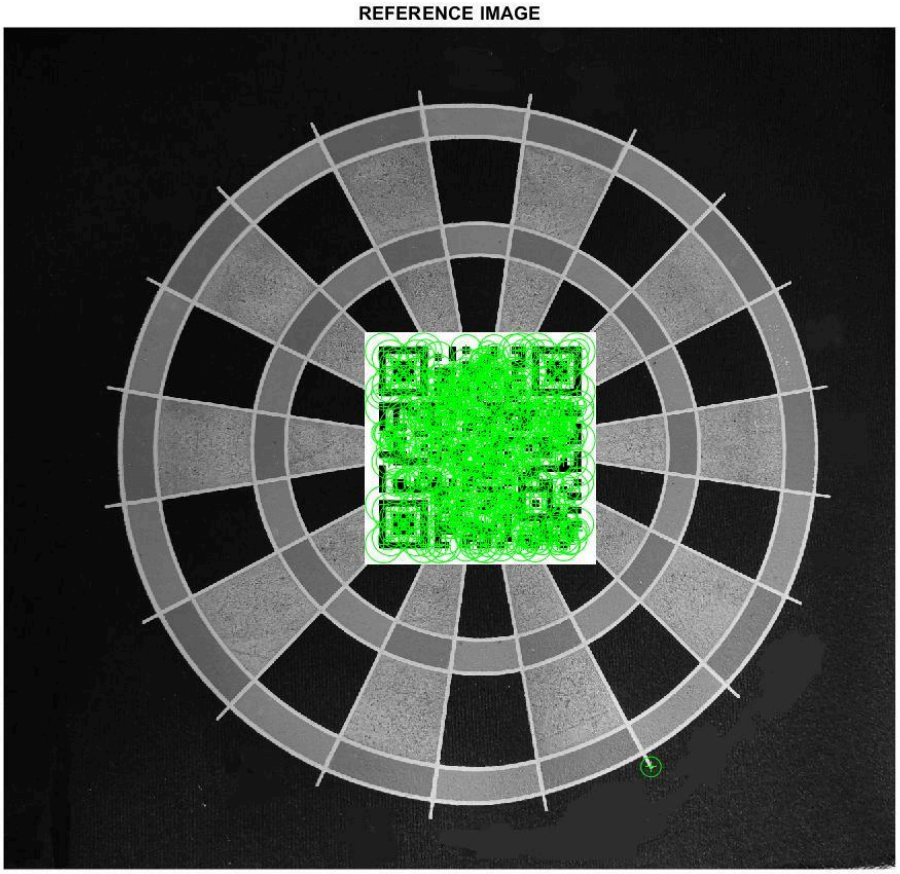

Matched Points in 1.2 scale

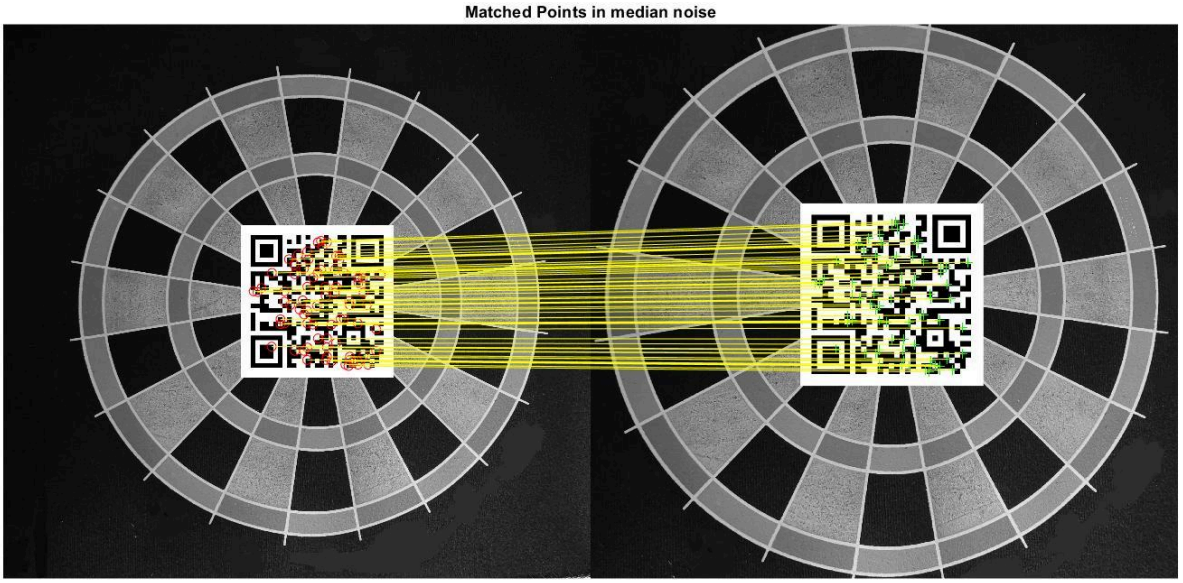
HARRIS

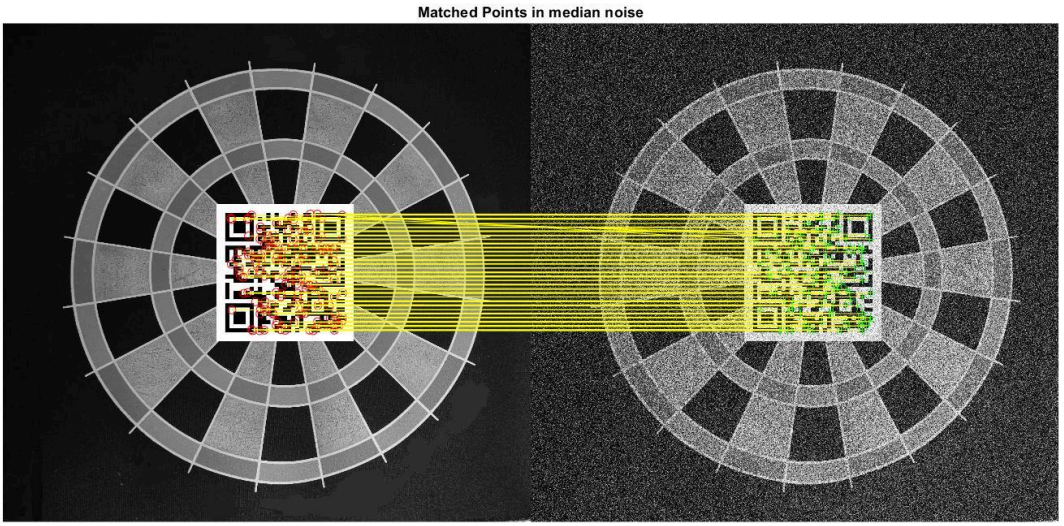




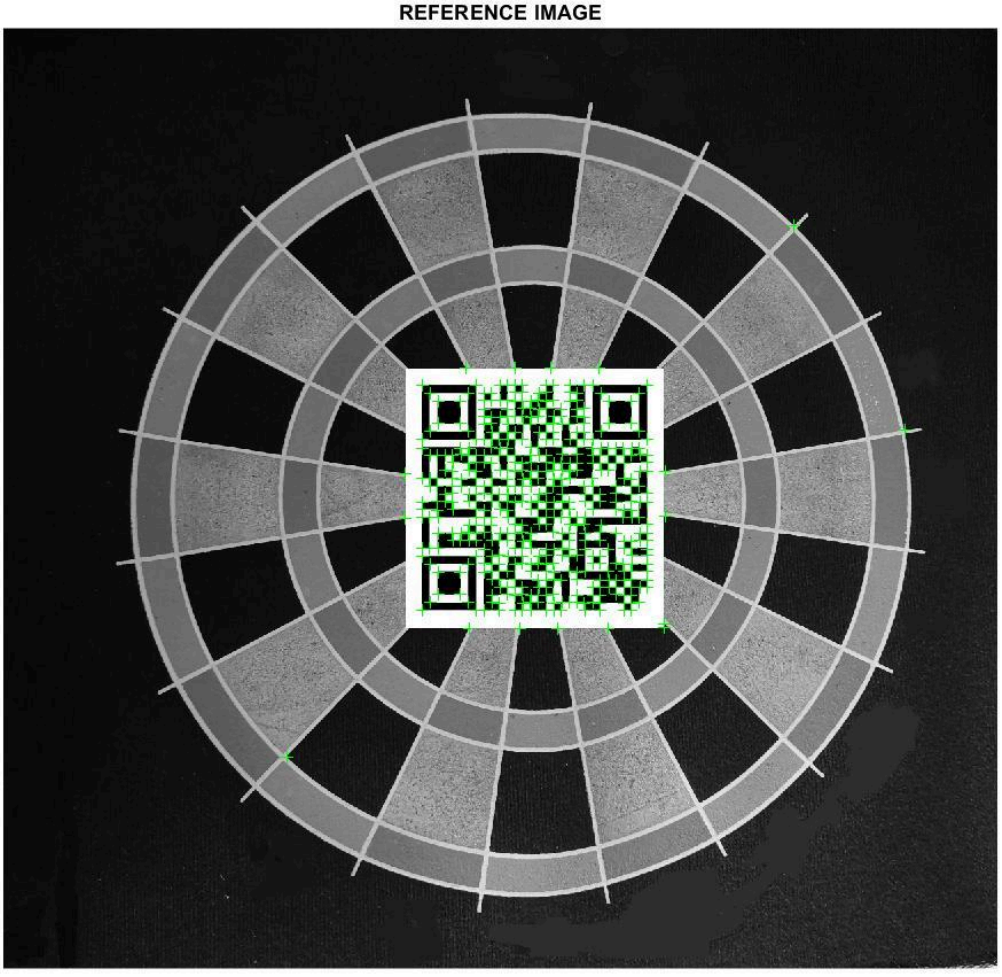
KAZE





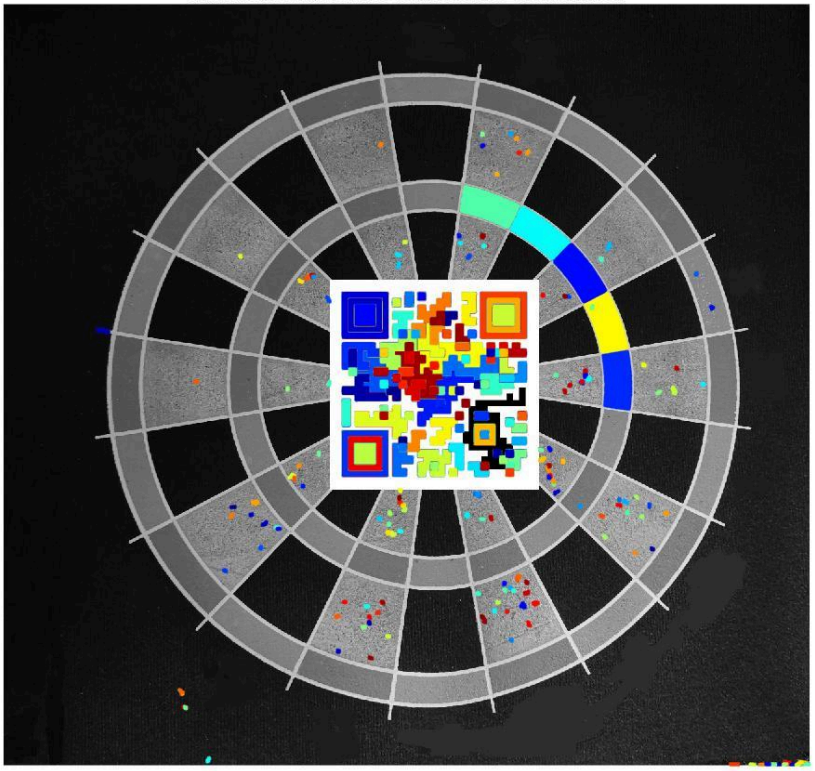


MINEIGEN

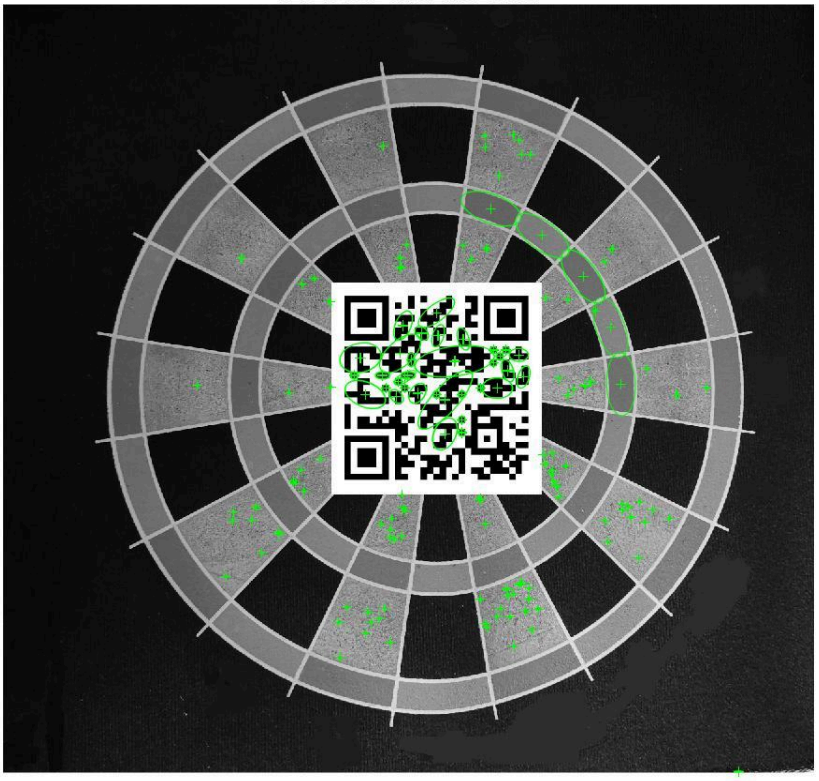


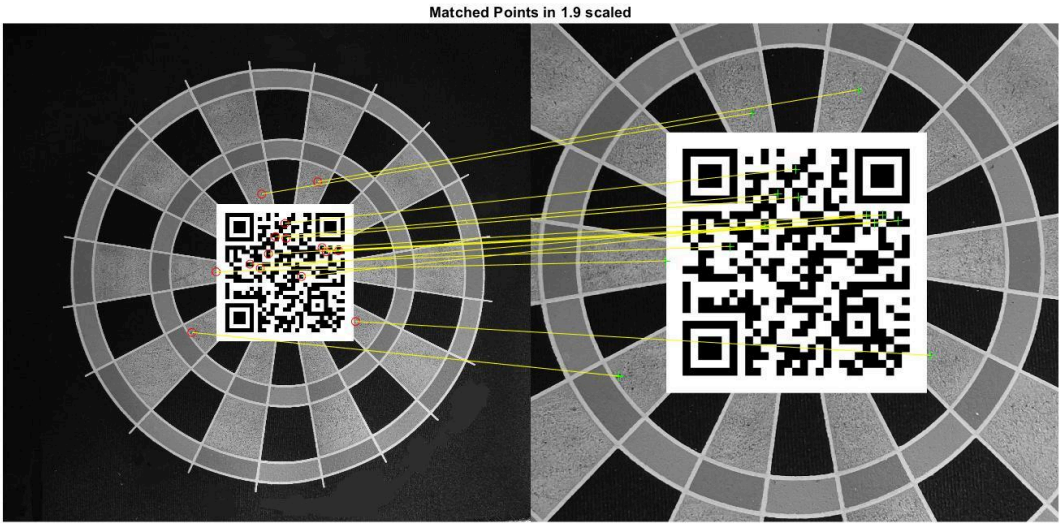
MSER

REFERENCE IMAGE MSER REGION IDENTIFICATION

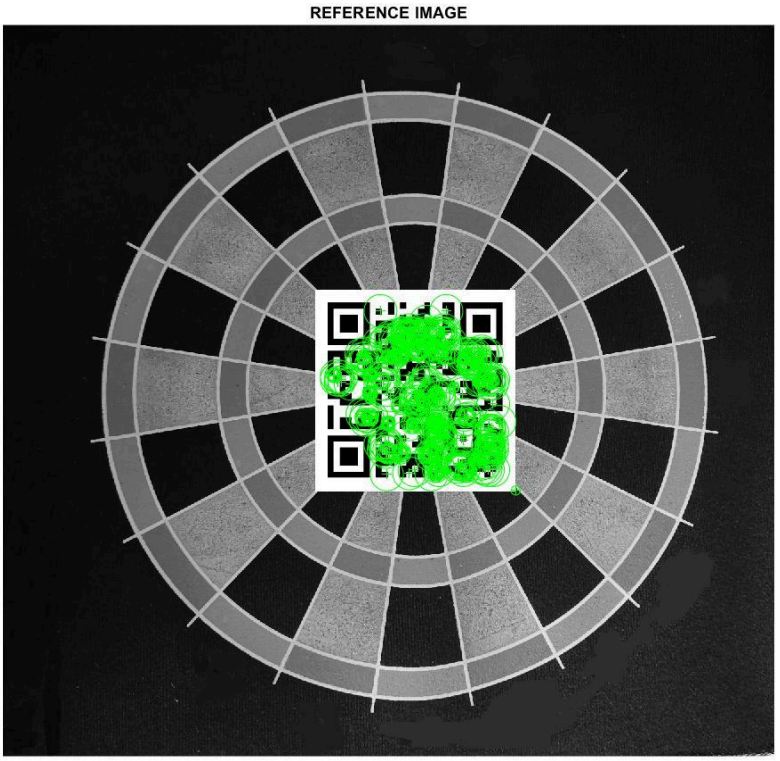


REFERENCE IMAGE PIXEL LIST

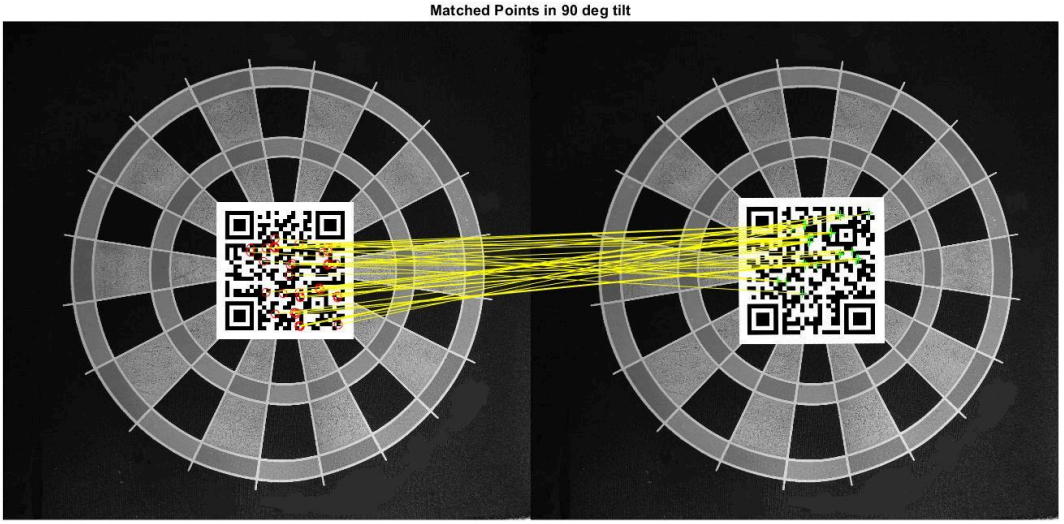




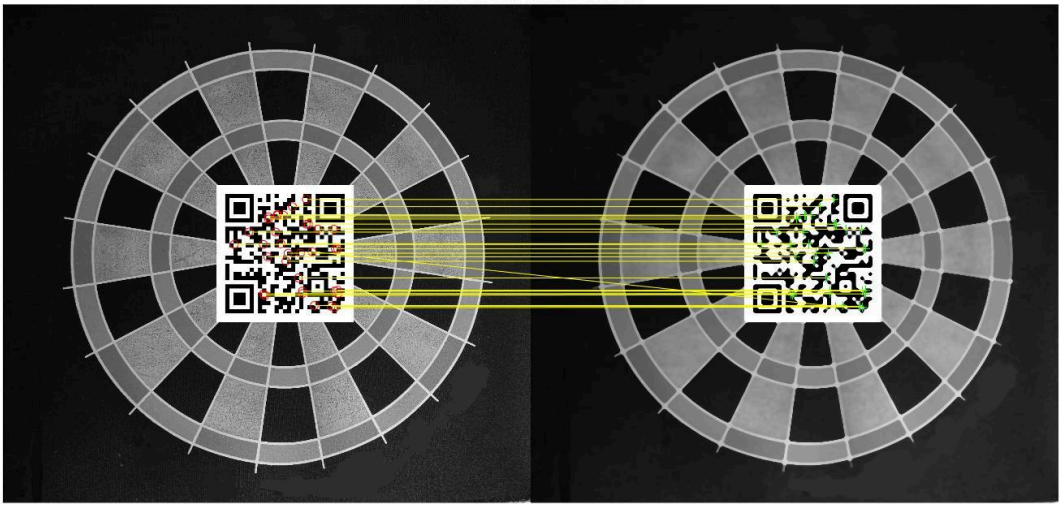
ORB



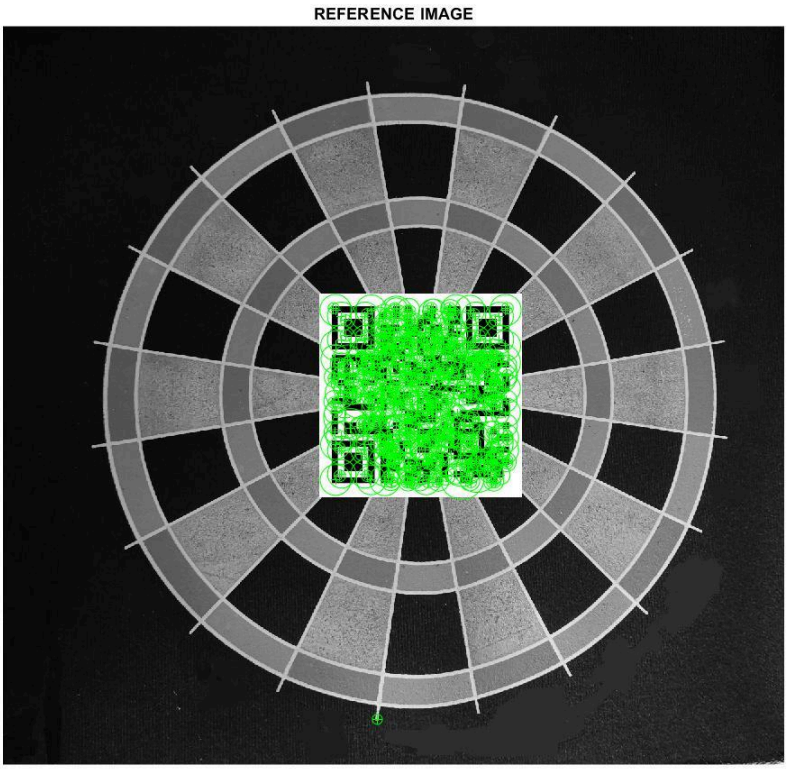
Matched Points in 90 deg tilt

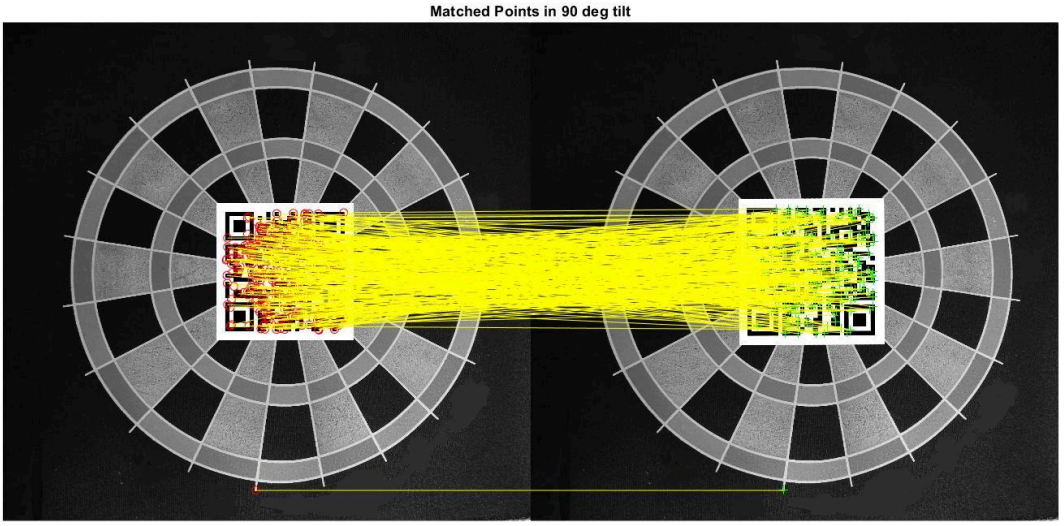


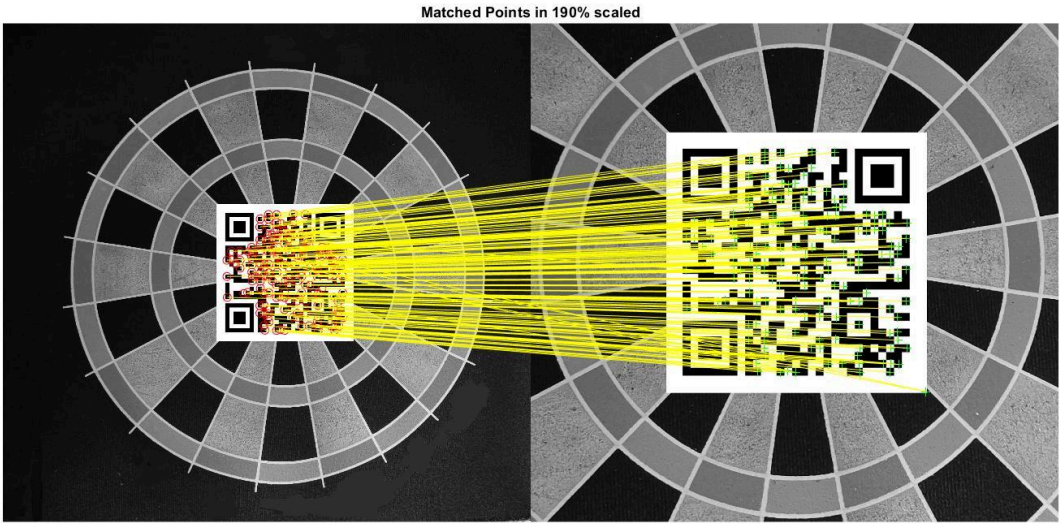
Matched Points in median noise



SURF

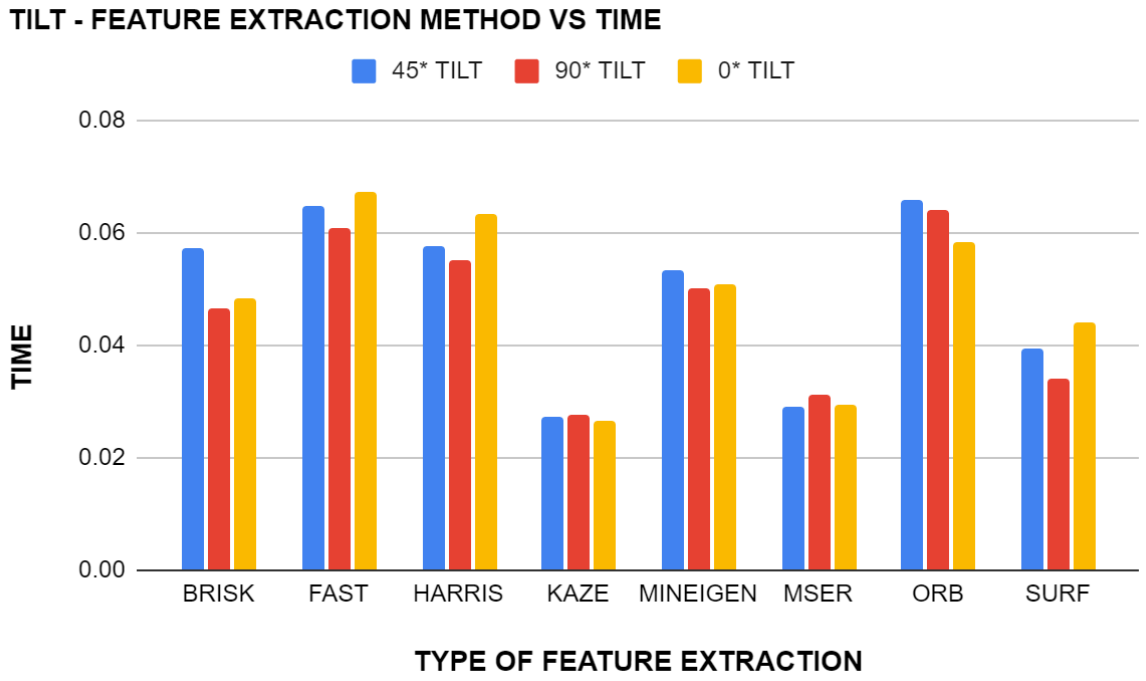





Matched Points in 190% scaled

TABULAR OBSERVATIONS AND GRAPHS

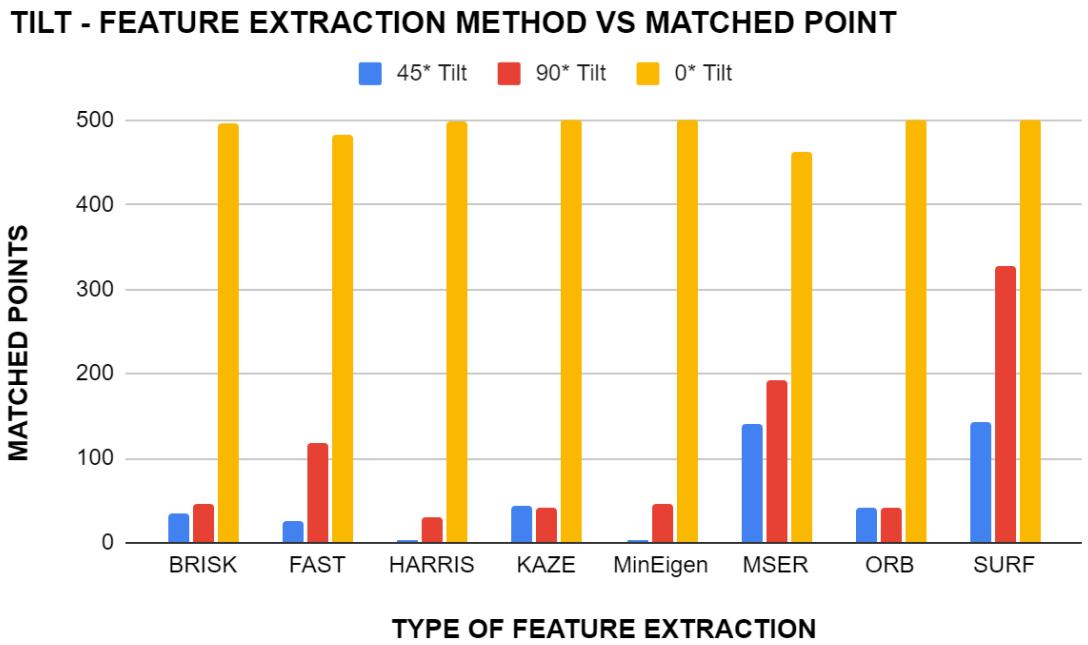
TILT - FEATURE EXTRACTION METHOD VS TIME			
ALGORITHM/ TILT ANGLE	45* TILT	90* TILT	0* TILT
BRISK	0.057429	0.046397	0.048213
FAST	0.064922	0.060837	0.067189
HARRIS	0.057784	0.055006	0.063457
KAZE	0.02712	0.027696	0.026556
MINEIGEN	0.053279	0.050198	0.05072
MSER	0.029002	0.031249	0.029311
ORB	0.06573	0.064043	0.05838
SURF	0.039339	0.03423	0.044163



The above graph is plotted between the different feature extraction algorithms and the time taken by these algorithms to cross-match the interest point in the reference image to the points in the variants of the reference image with different tilts of 0°,45° ,90°.As visible from the graph KAZE algorithm was able to cross-match all the three cases of tilt with the reference image with in a time range of 0.026 to 0.027 seconds.It can also be observed that the time taken by KAZE to cross-match is almost consistent for all the three type of tilt image inputs whereas methods like ORB show large variation time taken for different tilts.Moreover the highest time of computation for the cross-matching process was observed in ORB and Fast methods

making these methods less suitable for scenarios which require high computational efficiency and quick outputs.

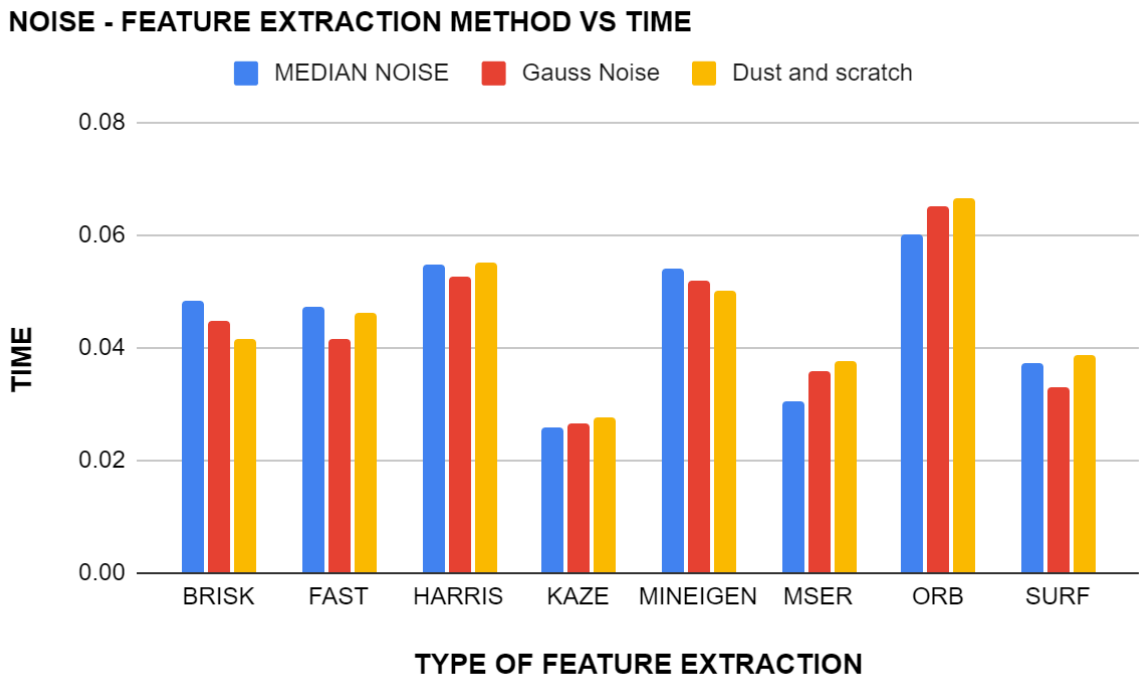
TILT - FEATURE EXTRACTION METHOD VS MATCHED POINT			
algorithm	45* Tilt	90* Tilt	0* Tilt
BRISK	35	47	496
FAST	25	118	483
HARRIS	3	31	498
KAZE	44	41	500
MinEigen	3	46	500
MSER	140	193	463
ORB	41	41	500
SURF	142	327	500



The above graph is plotted between the different feature extraction algorithms and the number of matched points.As evident from the graph, the number of match points is maximum in 0° tilt for all the different types of feature extraction.Comparing the above two graphs of tilt input scenario it can be concluded

that ORB requires higher computational infrastructure compared to methods like KAZE and MSER as the latter method was able to extract approximately same number of interest points with in one by third fraction of time consumed by ORB and FAST methods.Also,in contrary to the prior traits SURF has cross-matched the maximum number of points in all three type of tilt inputs.

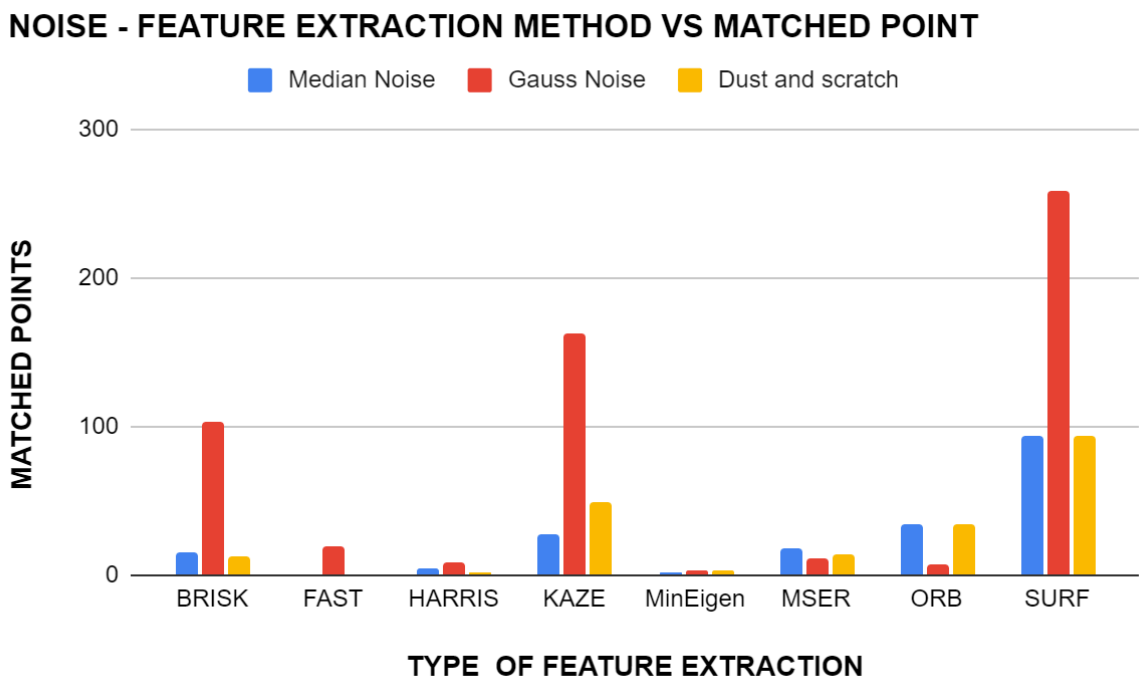
NOISE - FEATURE EXTRACTION METHOD VS TIME			
ALGORITHM	MEDIAN NOISE	Gauss Noise	Dust and scratch
BRISK	0.048426	0.044872	0.041388
FAST	0.047351	0.041478	0.04614
HARRIS	0.054686	0.05249	0.05528
KAZE	0.025917	0.026505	0.027484
MINEIGEN	0.054124	0.051961	0.050279
MSER	0.030573	0.035674	0.037759
ORB	0.060046	0.065243	0.066454
SURF	0.037223	0.033108	0.038761



The graph has been plotted for noise-feature extraction of three different types of noise against the time taken for these extraction by each method. From the graph, we can clearly observe that KAZE took the least time, between 0.0259sec (for median noise) and 0.0274sec (for Dust and Scratch) for feature extraction for all three types of noise, followed by MSER and SURF. The time taken by KAZE is also consistent

when compared to the other methods where the time taken differs from each other by a greater value. We can also observe that ORB is the method that took the longest for extracting the features for all three types of noises and thus making it the least suitable among the bunch if the images to be cross-matched has a significant amount of noise.

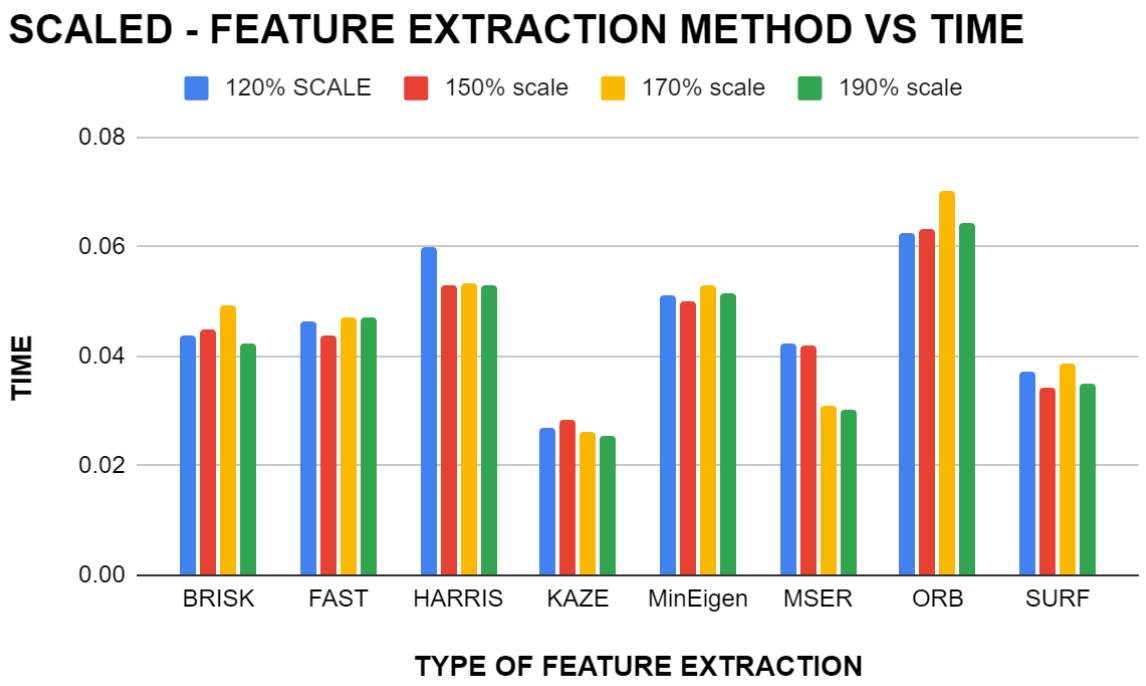
NOISE - FEATURE EXTRACTION METHOD VS MATCHED POINT			
algorithm	Median Noise	Gauss Noise	Dust and scratch
BRISK	15	103	13
FAST	0	20	1
HARRIS	4	9	2
KAZE	28	162	49
MinEigen	2	3	3
MSER	18	12	14
ORB	34	8	34
SURF	94	259	94



The graph has been plotted for noise-feature extraction of three different types of noise against the number of matched points they were able to come up with for the different method. It can be inferred from the graph that SURF has the highest number cross-matched points with Gaussian noise having the highest number

among the different noises. KAZE and BRISK are the methods that have some comparable stats and all the other methods falling behind them by a large margin. So when it comes to extracting the feature points from the images with noise, SURF is the best. If we are to combine time taken and the number of points they were able come up with in that period, SURF is the best available choice for cases where noise is there.

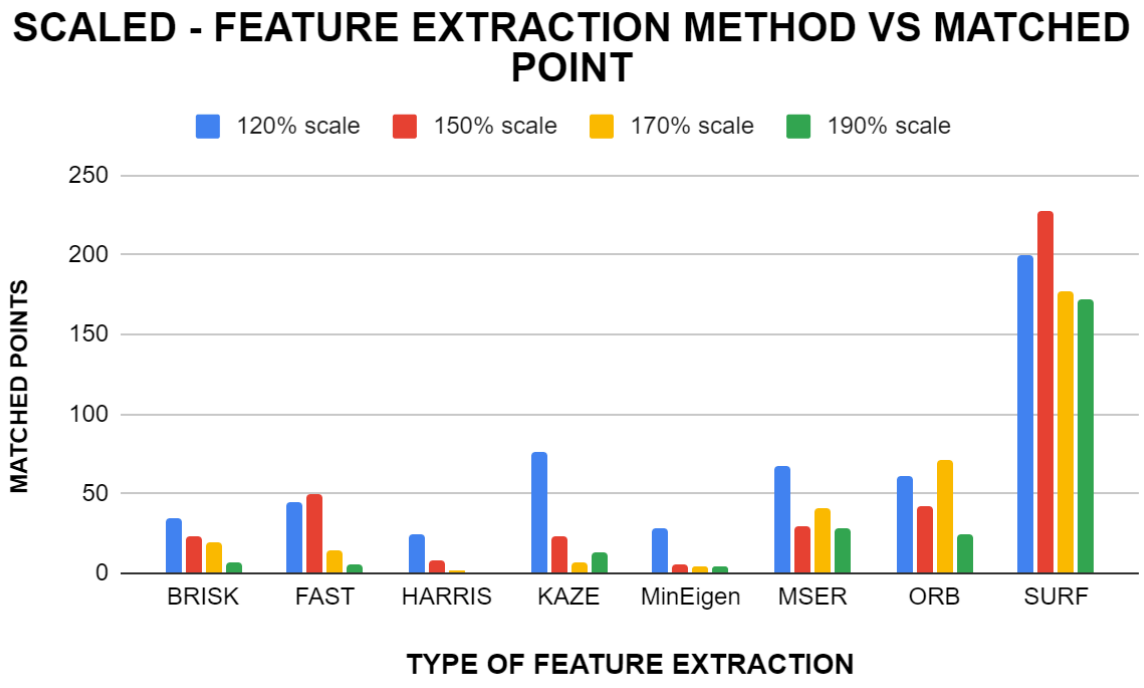
SCALED - FEATURE EXTRACTION METHOD VS TIME				
ALGORITHM/ SCALED FACTOR	120% SCALE	150% scale	170% scale	190% scale
BRISK	0.0439	0.04487	0.049317	0.042299
FAST	0.046347	0.043665	0.047194	0.047015
HARRIS	0.059786	0.052826	0.053238	0.053117
KAZE	0.026761	0.028547	0.026327	0.025562
MinEigen	0.051124	0.049958	0.052885	0.051355
MSER	0.042234	0.041882	0.030799	0.030086
ORB	0.062399	0.063155	0.070053	0.064498
SURF	0.037279	0.034314	0.038728	0.035153



The above graph is plotted between different feature extraction algorithms and time taken by these algorithms for feature extraction.The scaled variants of the Nrefrence image was provided as the input for plotting the graphs. It is evident from the graph

that the KAZE method has the least computation time ranging from 0.026 to 0.028 whereas the ORB method requires the maximum computation time ranging from 0.062 to 0.070.It is also noteworthy that the SURF method maintained its trait by requiring a computation time slightly above the method with least computation time.

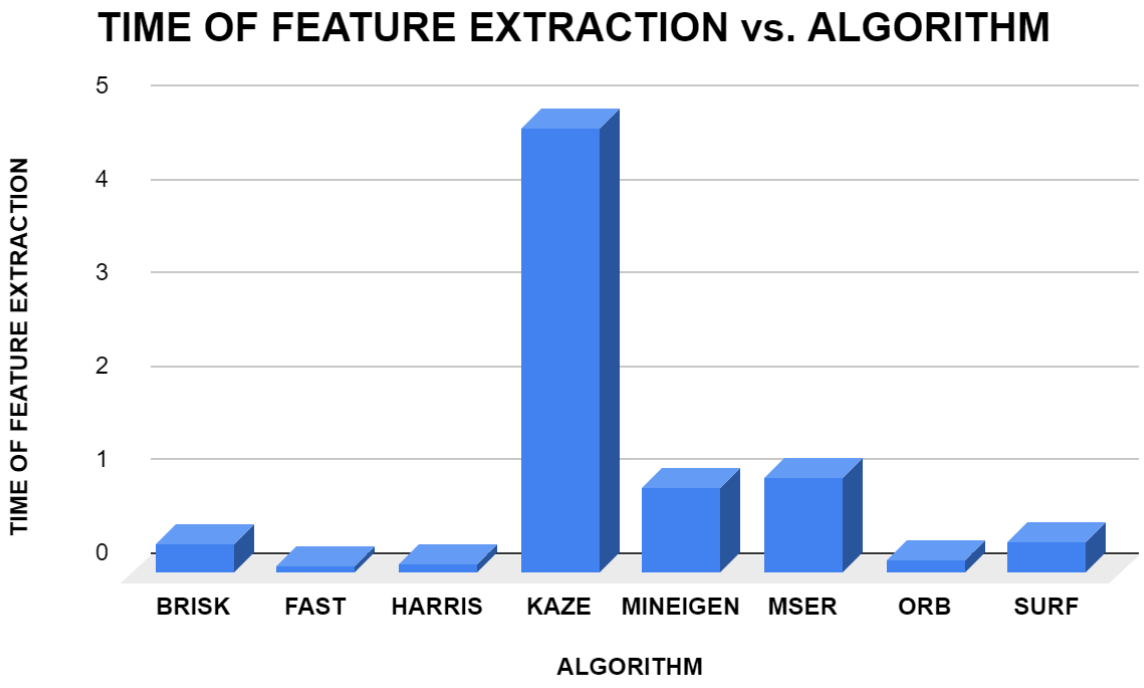
SCALED - FEATURE EXTRACTION METHOD VS MATCHED POINT				
algorithm	**120% scale**	**150% scale**	**170% scale**	**190% scale**
BRISK	**35**	**23**	**19**	**7**
FAST	**45**	**50**	**14**	**6**
HARRIS	**25**	**8**	**2**	**0**
KAZE	**76**	**23**	**7**	**13**
MinEigen	**28**	**5**	**4**	**4**
MSER	**67**	**29**	**41**	**28**
ORB	**61**	**42**	**71**	**25**
SURF	**200**	**228**	**177**	**172**



The above figure shows the graph between different feature extraction algorithms and the maximum number of matched points of these algorithms.Scaled image variants of the reference image was provided as the input in above graph and the results were as expected with SURF uncovering maximum number of match

points.Although KAZE method required the least amount of computation time (0.026-0.028) it was only able to locate relatively very few matchpoints.
As evident from the above two graphs, SURF was able to locate the maximum number of match points, within a comparatively lesser amount of time. Thus it can be concluded that for scaled input variants SURF has the maximum computational efficiency.

ALGORITHM	TIME OF FEATURE EXTRACTION
BRISK	0.313587
FAST	0.074147
HARRIS	0.0853465
KAZE	4.763558
MINEIGEN	0.91857
MSER	1.010411
ORB	0.148321
SURF	0.334866



The graphs is plotted between different types of feature extraction algorithms and the time taken by these algorithms to extract the feature points from a given

reference image.It is evident from the graph that the highest time for feature extraction was for KAZE and the lowest time was recorded for FAST feature extraction method. Although KAZE required higher computation time, the number of matched points were comparatively higher for KAZE ,emphasising on the linear relation between computation time and number of matched points.From the above visualisation it can observed that the SURF feature extraction method requires comparatively less computation time in all variants of inputs and provides a relatively higher number of match points in most of the cases.Thus it can be concluded that SURF is befitting in situation where the raw inputs are provided.